74

Sommario

RUBRICHE

Direttore: Alessandro de Simone
Coordinatore: Marco Miotti
Redazione/collaboratori: Mauro Lussignoli, Luca Viola, Carlo D'Ippolito, Donato De Luca, Paolo Agostini, Davide Ardizzone, Claudio Baiocchi, Luigi Callegari, Sergio Camici, Umberto Colapicchioni, Laura & Miria Colombo, Valerio Ferri, Simona Locati, Michele Maggi, Giancarlo Mariani, Clizio Merli, Marco Mietta, Marco Miotti, Oscar Moccia, Roberto Morassi, Guido Pagani, Antonio Pastorelli, Domenico Pavone, Armando Sforzi, Sonja e Patrizia Scharrer, Dario Pistella, Fabio Sorgato, Valentino Spataro, Danilo Toma, Franco Rodella, Stafano Simonelli
Direzione, pubblicità: Via Mosè 22 - 20090 Opera (MI) - Tel. 02/55500310 - Fax 02/57603039 - Redazione: Tel. 02/55500310
Pubblicità: Milano: Leandro Nencioni (direttore vendite), - Via Mosè 22 - 20090 Opera (MI) - Tel. 02/55500310
● **Emilia Romagna:** Spazio E - P.zza Roosevelt, 4 - 40123 Bologna - Tel. 051/236979
● **Toscana, Marche, Umbria:** Mercurio srl - Via Rodari, 9 - San Giovanni Valdarno (AR) - Tel. 055/947444
● **Lazio, Campania:** Spazio Nuovo - Via P. Foscari, 70 - 00139 Roma - Tel. 06/8109679
Abbonamenti: Liliana Spina
Arretrati e software: Opera, Via Mosè. 22 - Tel. 02/55500310 - Sig.ra Lucia Dominoni (il servizio è operativo nelle ore pomeridiane.)
Tariffe: prezzo per copia L. 6000. Abbonamento annuo (11) fascicoli L. 60.000. Estero il doppio.
Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 100.000
I versamneti vanno indirizzati a: SYSTEMS EDITORIALE Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizione, impaginazione a video e fotolito: Systems Editoriale Srl.
Stampa: Systems Editoriale/La Litografica Srl - CUGGIONO (MI)
Registrazioni: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70%
Distributore.: Parrini - Milano
Periodici Systems: Banca Oggi - Commodore Club (disco) - Commodore Computer Club - Commodore Computer Club (disco produzione tedesca) - Computer - Computer disco - Electronic Mass Media Age - Hospital Management - Jonathan - Nursing '90 - Pc Programm (disco) - Personal Computer - Software Club (casseta ed. Italiana) - TuttoGatto - Videoteca - VR Videoregistrare

IL TELEFONO, LA "SUA" VOCE

Le opportunità tecnologiche che oggi l'informatica mette a disposizionec onsentono notevoli comodità; anzi, le scomodità sono tangibili se un servizio telematico viene meno...

di Alessandro de Simone

Giorno piovoso di aprile. Sto preparando gli ultimi articoli di *Commodore Computer Club*, anche se ho mal di testa e mi sento le ossa rotte: devo avere la febbre.

Telefono a Marco Miotti dicendo che, il giorno dopo, preferisco non venire in Redazione per non aggravare il mio stato di salute e lo prego di mettersi in contatto, l'indomani mattina, per ricevere, via modem, gli ultimi articoli; li terminerò prima di andare a letto.

Fino a tardi clicko, salvo, stampo, correggo, imposto.

L'ultimo file di Ventura viene registrato intorno alla mezzanotte, mentre fuori infuria un temporale.

Il mattino dopo tento di mettermi in contatto con la nostra BBS.

Tento è un termine inadatto: in effetti impreco, mi arrabbio, urlo e guardo lo schermo del mio PC che non fa altro che inviare, e ricevere, sporcizie di ogni tipo.

Un successivo contatto "vocale" con la Redazione mi conferma che, purtroppo, la linea è "sporca", forse a causa di qualche contatto dovuto alla pioggia incessante, forse perchè è ora di punta, forse perchè, molto più banalmente, siamo nelle mani della SIP.

Non mi rimane che coprirmi bene, caricare i dischetti, infilarmi in macchina e recarmi ad Opera; 30 chilometri per andare ed altrettanti per tornare: tempo, benzina e pazienza persi inutilmente per le linee "sporche".

Non ho riferito l'episodio per lamentarmi, un'ennesima volta, dei servizi SIP, ma solo per far riflettere il lettore sull'attualità tecnologica che abbiamo la fortuna di vivere. Fino a poco tempo fa era impensabile effettuare un servizio per telefono, nè era immaginabile svolgere un lavoro stando seduti comodamente a casa propria ed esser costretti ad uscire solo in caso di estrema necessità.

L'utente informatico attuale non può fare più a meno della telematica, dei suoi servizi, delle sue opportunità di scambio dati, programmi ed informazioni in generale. Un ulteriore invito ad interessarsi dei modem, quindi, giunge più che mai opportuno, soprattutto tenendo conto del notevole abbassamento dei costi di produzione e della conseguente diffusione dei modem.

SIP permettendo...

DIVISILLABE

Da un po' di tempo vedo che C.C.C. riporta numerosi errori di battitura riguardanti la divisione in sillabe. Possibile che i redattori non si accorgano dei vistosi errori in fase di correzione delle bozze?
(da alcune lettere)

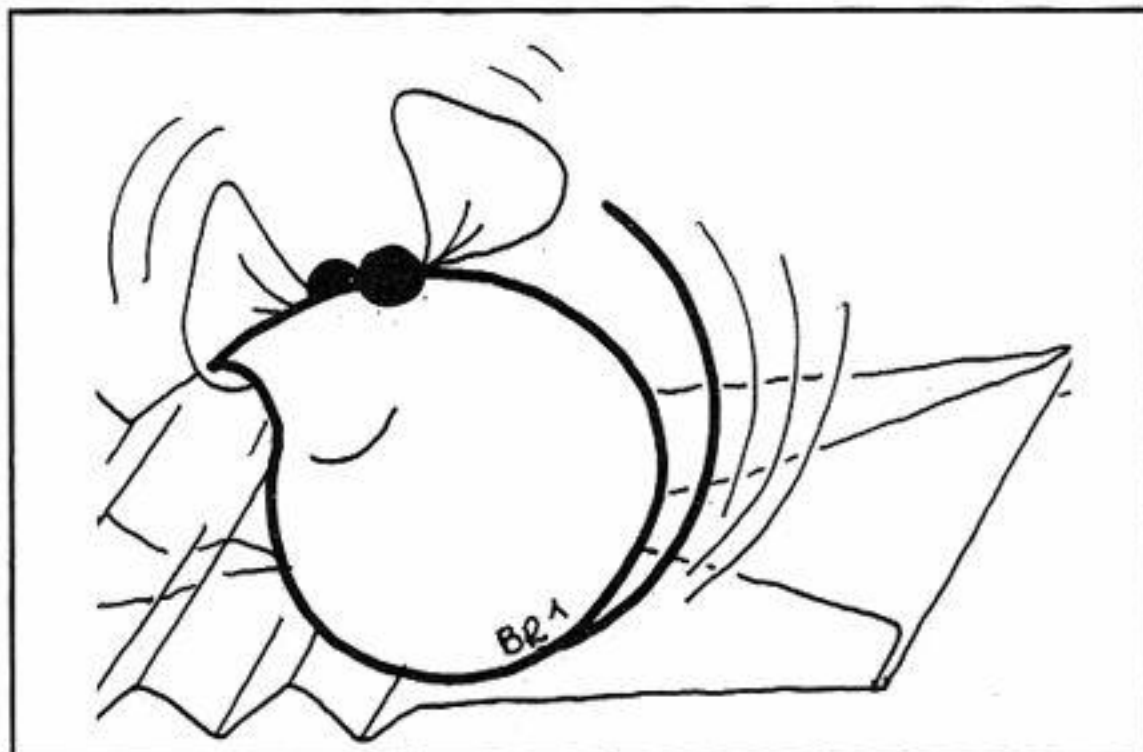
Gli errori non sono dovuti nè agli autori degli articoli nè ai redattori. Il vero ed unico responsabile è, infatti, il sistema di videoimpaginazione che, benchè settato sulla lingua italiana, spesso e volentieri genera svarioni di ogni tipo.

Durante l'impaginazione, in verità, non appena ci accorgiamo degli errori provvediamo a *"forzare"* la divisione in sillabe; purtroppo capita che, per esigenze di incolonnamenti, alcune lunghezze (di righe o di colonne) debbano essere in seguito reimpostate con la conseguente reimpostazione globale ed automatica dell'intero articolo.

Alcune parole che prima dell'operazione erano intere, pertanto, vengono stavolta divise in sillabe secondo l'errato algoritmo ed è praticamente impossibile accorgersene e porre rimedio.

Il nome del pacchetto D.T.P. che usiamo? Meglio tacere...

MODEM 6499

Posseggo un adattatore telematico 6499, ma non dispongo di software idoneo ad un suo uso più sofisticato.
Potete aiutarmi a rintracciarlo, pubblicando, magari, il mio indirizzo in modo da realizzare una "rete" di utenti 6499?
Francesco Passera
Via Ivrea, 8
10017 Montanaro (To)

Il modem 6499 non è una tavoletta grafica che ha bisogno indispensabilmente di software per funzionare.

Al suo interno, infatti, dispone già della *Rom* contenente il programma per farlo funzionare. A questo punto basta formare il numero di telefono e scambiar dati.

In effetti il 6499 non è l'optimum della comunicazione telematica, soprattutto considerando l'attuale produzione di modem. Quando, però, l'adattatore telematico fu lanciato dalla Commodore (diversi anni fa) era uno dei modelli più incredibili ed a basso prezzo che era possibile procurarsi.

Come vedi, comunque, non ho difficoltà a pubblicare il tuo annuncio. Ti consiglio, però, di pensare seriamente a cambiar computer dal momento che ciò che, oggi, risparmi non acquistando un elaboratore più potente, lo spendi in bollette Sip a causa della modesta velocità di gestione dell'apparecchio.

FILE DATA ERROR

Cercando di caricare un file da nastro, con il mio C/64, compare il messaggio "File data error". Che cosa vuol dire?
(Nunzio Labranca)

Purtroppo non hai inviato nè programma nè file e non comunichi nemmeno di che tipo di dati si tratta.

Ritengo, pertanto, che tu sia un principiante che, con

E PER AMIGA?

Nelle precedenti confezioni di Commodore Computer Club abbiamo inserito un dischetto, in omaggio, del formato 5.25 pollici. Dal momento che il dischetto è incompatibile con Amiga, gli utenti di questo computer hanno protestato presso la Redazione. Per venire loro incontro si è quindi deciso di lanciare l'offerta speciale "Quattro al prezzo di due". In altre parole gli utenti Amiga potranno richiedere, secondo le solite modalità (riportate in altre pagine del presente fascicolo) quattro fascicoli arretrati, oppure quattro dischetti della serie Amigazzetta, pagando l'importo relativo a due soli numeri (oppure a due soli dischetti).

molta buona volontà, cerca disperatamente di supplire alle *notevoli* carenze del manuale allegato alla confezione del computer.

Una trattazione molto approfondita è stata affrontata su molti fascicoli di C.C.C.

Ti consiglio di studiare, soprattutto, i due inserti pubblicati in precedenza: mi riferisco a *"Come gestire senza problemi i file sequenziali"* (C.C.C. n. 39); *"...e non indurci nei Syntax error"* (C.C.C. n. 31); *"Gli errori dovuti alla manipolazione delle stringhe"* (C.C.C. n. 32). I tre argomenti citati occupano, ciascuno, l'intero inserto del corrispondente fascicolo cui si riferiscono.

C/64, VITA MEDIA

Qual è la vita media di un un C/64 tenuto, mediamente, acceso 5 - 6 ore al giorno?
(Angelo Rinaldi - Castellana)

Un computer moderno, come un qualsiasi apparecchio elettronico attualmente prodotto, dovrebbe offrire una vita media praticamente infinita.

In realtà, tuttavia, qualcosa può accadere a qualche minuscolo chip al silicio che, come ogni cosa terrena, non può essere eterno.

Nè si può immaginare di attribuire vite più lunghe a computer di marca più prestigiosa.

Il mio *C/128-D*, tanto per far un esempio, è uno dei primi esemplari prodotti dalla Commodore, gira egregiamenete da molti anni e non mi ha mai dato fastidio.

Invece uno dei mie computer *Ms-Dos* (che, in teoria, dovrebbe risultare più affidabile del "giocattolo" C/128 sia perchè espressamente *"professionale"*, sia perchè di marca "prestigiosa") mi sta dando seri problemi di Ram e dovrò, quanto prima, portarlo a riparare.

Pensa a campare, quindi, e nel frattempo tocca ferro...

MPS-1230

Ho acquistato una Mps-1230, ma non riesco a trovare un software di scrittura in grado di farla funzionare.
(Nevio Tavoni - Pescara)

Ma scherziamo? Un qualunque sistema di videoscrittura (*Easy Script, Geos* eccetera) per C/64 è in grado di gestire la stampante citata.
E' molto probabile che le istruzioni del software in tuo possesso siano incomplete (o che l'apparecchio sia difettoso!).

Procurati i libretti di istruzione originali e vedrai che la 1230 filerà come un treno.

MANUALE DI RIFERIMENTO

Spesso, nel manuale di istruzioni allegato al C/64, si fa riferimento ad un fantomatico "Manuale di riferimento del programmatore".
Vorrei sapere se è in italiano e dove è possibile acquistarlo.
(Emanuele Giacometti - Venezia)

Il volume, edito dalla stessa Commodore, è in lingua italiana ed è formato da una miriade di capitoli relativi al basic, al linguaggio macchina, alla grafica, al suono, all'hardware, eccetera.

E' uno dei volumi più completi che esistano, ma altrettanto difficile da trovare se non presso librerie *molto* specializzate o per corrispondenza (hai provato a chiedere ai nostri inserzionisti che pubblicano le loro pagine pubblicitarie?).

SALVAVIDEO H/W

Vorrei commentare alcune parti del progetto "Salvavideo hardware" (C.C.C. n. 72).
Vi si afferma che il transistor BC-109 è usato per questioni di sicurezza, allo scopo di evitare un contatto diretto relè - computer; ritengo, al contrario, che il contatto ci sia egualmente perchè uno dei due capi è in contatto diretto con la porta tape del C/64. C'è da considerare, inoltre, che il relè è formato da una bobina avvolta su di un nucleo magnetico. Ad ogni apertura o chiusura dei contatti la corrente circola, generando, ai morsetti della bobina, una forza elettromotrice auto-indotta che tende ad opporsi alla causa che l'ha generata. Tale sovratensione, direttamente proporzionale alla velocità con cui varia la corrente, può distruggere il transistor presente nel computer. In simili casi, quindi, è bene porre un diodo sui morsetti del relè che sarà polarizzato inversamente (messo al contrario) proprio perchè agisca solo nei transitori. Un fusibile, posto a valle del relè, completerebbe la "sicurezza" del circuito.
(Andrea D'Orsi - Udine)

BUFFER DI TASTIERA

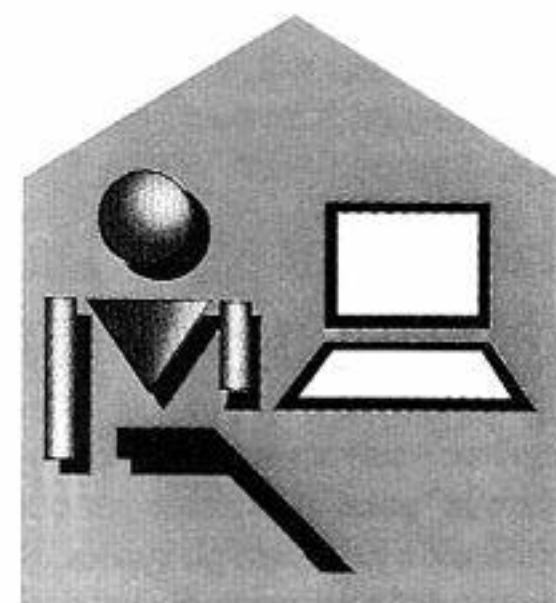
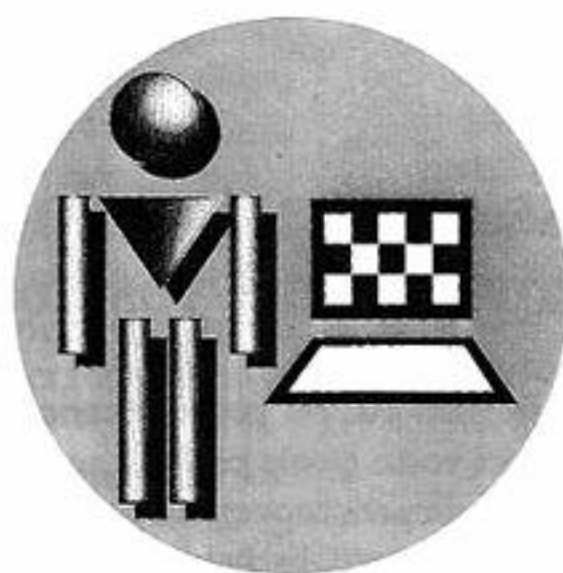
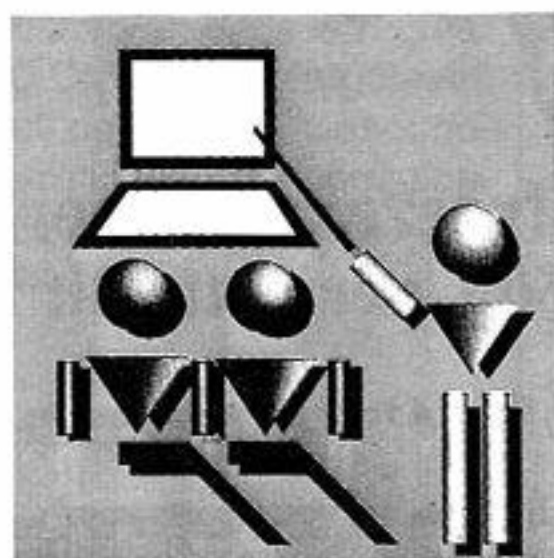
Come posso modificare il buffer di tastiera (locazione 649) in modo da gestire più di 10 caratteri?
(Ettore Amanranti - Roma)

Non è possibile, dal momento che l'operazione rischierebbe di gestire anche altre locazioni che il computer utilizza per altri scopi.

Ma perchè i 10 caratteri non ti bastano? Se avessi comunicato il problema che realmente vuoi risolvere avrei potuto darti una mano; è probabile, infatti, che altre tecniche, estranee alla manipolazione del byte 649, possano rendere ottimale la procedura che intendi portare a termine.

Per quanto riguarda le altre domande contenute nella lettera, è stata data una risposte in più di una occasione.

L'INFORMATICA PER

LO STUDIO L'HOBBY LA CASA

HA FINALMENTE UNA SUA MOSTRA-MERCATO

SYSTEMS EDITORIALE PER TE

La voce III
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far *parlare* il computer, sia di farlo *Cantare!* Diversi esempi allegati.
Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.
Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.
Cassetta: L. 12000
Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).
Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese ed i guadagni di una famiglia.
Cassetta: L. 12000
Disco: L. 12000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.
Cassetta: L. 12000
Disco: L. 12000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.
Cassetta: L. 20000
Disco: L. 20000

Analisi di Bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.
Cassetta: L. 20000
Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 ed i rudimenti di programmazione. Interattivo.
Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.
Cassetta: L. 12000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica: ideale per far avvicinare i bambini al calcolatore. Diversi esempi allegati.
Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.
Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.
Solo su disco: L. 25000

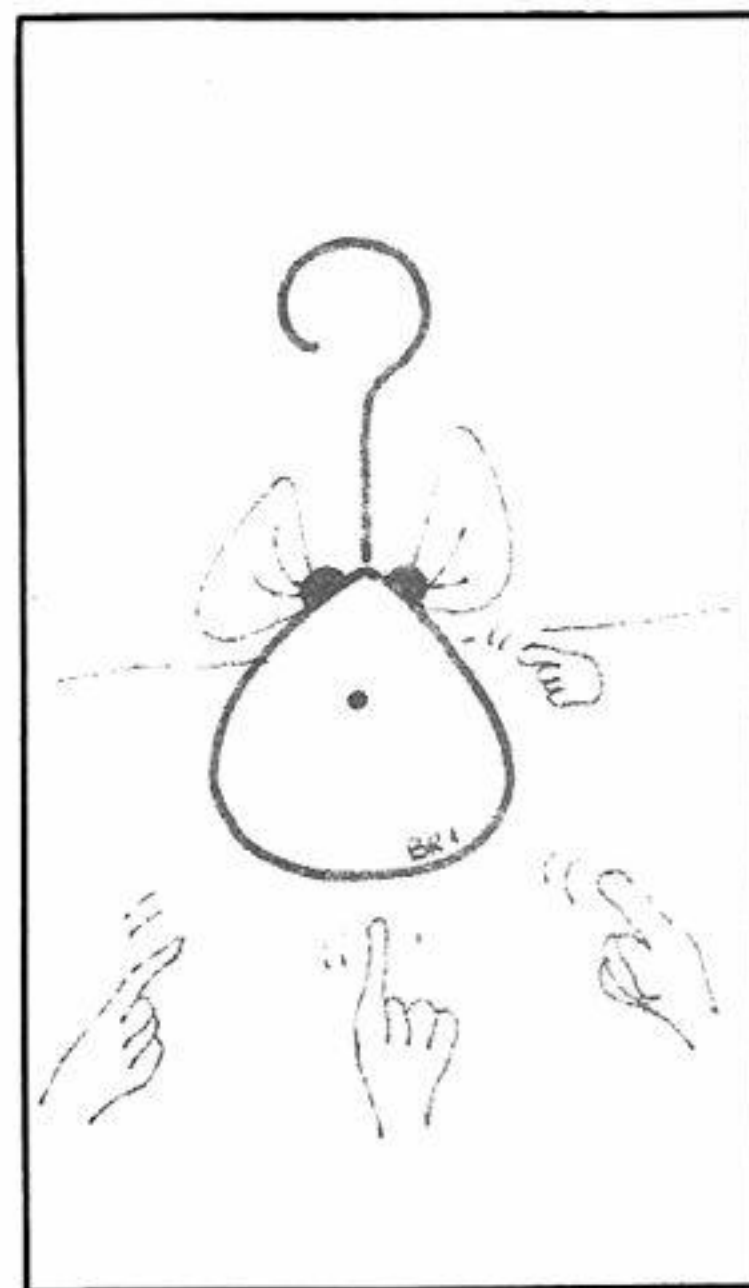
Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive. Ideale per studenti.
Disco: L. 25000

L.M. + Routine grafiche
Un fascicolo speciale (corredato di dischetto) suddiviso in due parti: corso completo di linguaggio macchina 6502 e implementazione di numerose routine che aggiungono al C/64 istruzioni Basic specifiche per la grafica, comprese quelle per disegnare in prospettiva!
Fascicolo + disco: L. 16000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.
Disco: L. 15000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.
Disco: L. 16000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.
Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su *"Commodore Computer Club"*. In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro. Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.
Ogni dischetto: L. 12.000

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/0 sia in formato Microsoft sia in vesione C/64 (da chiedere, a parte, su disco). ***(125 pag.)***

L. 7000

Dal Registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. ***(94 Pag.)***

L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. ***(112 pag.)***

L. 5000

Utilities e giochi didattici
Raccolta di numerosi programmi, in versione C/64 e Spectrum, di particolare interesse per chi intenda svliluppare software didattico. ***(127 pag.)***

L. 6500

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. *(127 pag.)*

L. 7000

Dizionario del Personal Computer
Raccolta dei termini più diffusi nel campo professionale; dizionario inglese - italiano. (Edizione ridotta). ***(96 pag.)***

L. 8000

Dizionario dell'Informatica
Dizionario inglese italiano di tutti i termini usati nell'informatica. (Edizione completa). ***(385 pag.)***

L. 20000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word Star, Samna, Multimate Advantage, Word 3. ***(79 pag.)***

L. 5000

Telefax
Volumetto divulgativo sull'importanza del Telefax e sulle sue modalità operative caratteristiche. ***(66 pag.)***

L. 5000

Come compilare un giornale aziendale in Azienda
I principali problemi per chi opera in ambiente DPT, affrontati e risolti con la massima chiarezza e semplicità. ***(80 pag.)***

L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza. ***(91 pag.)***

L. 5000

ABBONAMENTO

Commodore Computer Club

11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato

di C.C.C. L. 6000

COME RICHIEDERE I PRODOTTI SYSTEMS

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.

Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale
Viale Famagosta, 75
20142 MILANO

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.

Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.

Volendo una spedizione in contrassegno è necessario anticipare la cifra di L. 10000 (diecimila), da inviare secondo le modalità prima indicate, indipendentemente dalla quantità di materiale richiesto, e da conteggiare, comunque, IN AGGIUNTA alla cifra risultante dall'ordine. (Si sconsiglia, pertanto, la richiesta di di prodotti in contrassegno)

Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

LA POSTA DEL C/128

(a cura di Domenico Pavone)

CHE TEMPI, RAGAZZI!

Lavorando con l'orologio interno del mio C/128, mi sono reso conto che un listato come questo...

10 TI$ = "000000"

20 H$ = Left$ (ti$, 2): M$ = Mid$ (ti$, 3, 2)

30 S$ = Right$ (ti$, 2): Print Chr$(19);

40 Print "ore:"; H$: Print "minuti:"; M$

50 Print "sec:"; S$: Sleep 1: Goto 20

...presenta un errato conteggio dei secondi. Spesso, infatti, viene saltato un secondo, passando direttamente (p. es.) da 8 a 10. Eliminando l'istruzione Sleep, tutto torna normale: vuol dire che Sleep non è compatibile con la variabile di sistema TI$?

(Luca Serlini - Castenedolo)

L'inconveniente non è legato a problemi di compatibilità, ma piuttosto ai tempi di esecuzione dell'interprete basic, in associazione ad una non totalmente affidabile precisione di *TI$*.

Più chiaramente: quando il programmino va ad eseguire l'istruzione *Sleep 1*, che interrompe le operazioni per un secondo, la variabile *TI$* continua ad aggiornarsi, incrementandosi di un secondo ad ogni "passaggio".

Prima dei vari *Print* che mostrano lo stato di TI$, va però considerato anche il tempo necessario all'interprete per eseguire (dopo il *Goto 20*) le assegnazioni alle variabili stringa, che, tra l'altro, risultano proprio tra le operazioni più lente del basic.

Certo si sta parlando di frazioni di secondo, ma, dopo un certo numero di iterazioni, la loro somma provocherà il "salto" lamentato.

A livello di decimi di secondo, inoltre, la variabile TI$ ed il comando Sleep non sono del tutto affidabili, aggiungendo qualche altro millesimo di ritardo o anticipo al già precario equilibrio del programma.

Nel prosieguo della missiva, il lettore propone in alternativa una specie di contatore interno al programma, per supplire a tale imprecisione.

Ma il rimedio non può ovviamente essere questo: per gli stessi motivi, l'orologio risulterebbe assolutamente non conforme alla realtà, anche in presenza di una scansione dei secondi apparentemente regolare. Se si ha l'esigenza di rendere compatibile *Sleep* (o qualunque altro tipo di temporizzazione basic) con la reale scansione del tempo, l'unica soluzione possibile è quella di ricorrere ai veri orologi hardware del computer, ovvero i due cosiddetti *TOD* (Time Of Day) implementati dai due chip *CIA (Complex Interface Adapter)*.

Per la cronaca, sono locati agli indirizzi 56328 - 56331 (Cia 1) e 56584 - 56587 (Cia 2) di banco 15 e dispongono anche di allarme (come una sveglia, per intenderci), ma una loro corretta gestione è possibile solo da linguaggio macchina, per di più complicato (o semplificato, dipende dai gusti) dalla notazione *BCD (Binario Codificato Decimale)*.

Il tutto, considerate le esigenze di compatibilità col basic, condito da una spruzzatina di Interrupt.

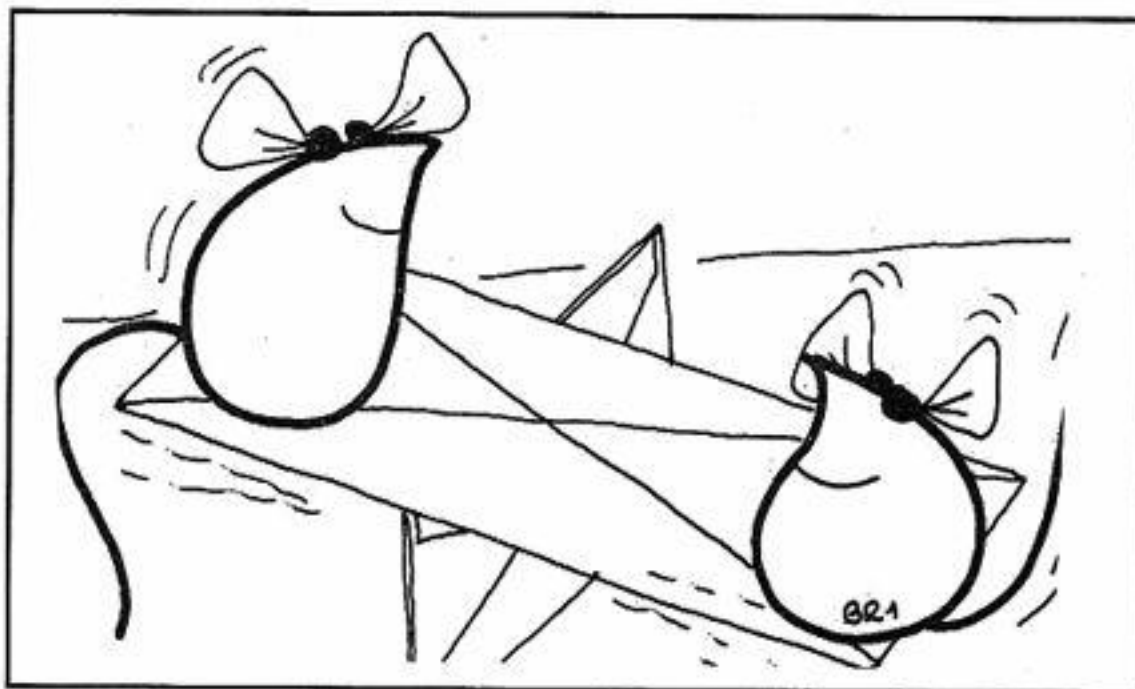
L'argomento, insomma, non è certo affrontabile in questa sede.

Se richiesto da più "voci" (triste esigenza, ahinoi, per un computer ormai in agonia), se ne potrà ugualmente parlare in qualche articolo espressamente dedicato al linguaggio macchina ed al Bcd in rapporto al C/128.

GEOS MUTILATO

Dopo aver sentito parlare con toni entusiastici del sistema Geos, mi sono decisa ad acquistarlo (V 1.5). Sul disco, però, non ho trovato i programmi che più mi interessavano: GeoWrite e GeoPaint. Dal manuale (in inglese) mi è parso di capire che i suddetti programmi non ci siano proprio, e siano da comprare separatamente. E' possibile?

(Elena Annuema - Milano)

Il pacchetto Geos, giunto peraltro alla versione 2.0 (salvo novità dell'ultima ora), comprende tanto il *GeoWrite* che il *GeoPaint*. E' probabile che l'acquisto sia errato, magari invece del Geos si tratta di qualche applicazione ad esso riservata. A meno che, ipotesi non del tutto da scartare, non sia una copia incompleta (vogliamo chiamarla pirata?) dell'originale...

(IN)UTILI AUTOBOOT

A proposito della risposta "Routine Fantasma" pubblicata nella posta per C/128 del n. 71, vorrei sapere se le stesse regole si possono applicare al C/128 in modo 64, e se si può far eseguire un programma basic tramite l'autoboot.
(Andrea - Vicenza)

L'argomento di *"Routine fantasma"*, in effetti, non trattava l'autoboot in generale, ma solo in rapporto alla possibilità di "nascondere" alla directory un eventuale programma.

In tal senso, una risposta alla seconda domanda è già esplicitamente inclusa in quelle righe.

Il semplice autoboot di un programma basic, è invece la cosa più facile di questo mondo, anche per chi non abbia voglia di approfondire l'argomento dal punto di vista programmazione: basta adoperare la routine basic *Boot Maker* inclusa nel dischetto fornito assieme al computer, che comprende un'opzione di scelta tra programmi basic ed LM. Il modo 64, in senso stretto, non consente l'autoboot.

Tuttavia, è teoricamente possibile applicare quanto esposto sul n. 71.

Nel lontano n. 41 (aprile '87 pag. 69), è stata infatti pubblicata una (utilissima) routine di autoboot che consente passaggio, caricamento ed esecuzione automatica di programmi in modo 64. Disponendo delle necessarie cognizioni *Assembly* e di molta, molta pazienza, si potrebbe anche modificarla per lanciare un programma "fantasma" memorizzato su tracce e settori di un disco. Ma, in tutta franchezza, a che pro?

Un simile programma girerebbe solo in modo 64, ma potrebbero usufruirne esclusivamente i possessori del C/128!

RIDOTTO, MA FORMATTA

Come si fa ad inserire in un programma, dopo un comando Header (a$), la ID di un disco da formattare adoperando una variabile?
(Alberto Bianchi - Roma)

Il manuale in dotazione al C/128, alla voce *Header*, riporta a chiare lettere una nota che precisa: *"Non è possibile utilizzare una variabile stringa come i.d."*.

Tuttavia, forzando un po' la mano, qualcosa si può fare.

Ecco la soluzione:

10 A$ = "miodisco": Id$ = "xx"
20 Header (a$) + Chr$ (44) + Id$

Si provi a mandare in esecuzione questo miniprogramma (badando che il disco inserito nel drive sia realmente da formattare), e tutto funzionerà regolarmente.

Dopo la formattazione, si avrà un disco con nome *Miodisco* e i.d. *xx*.

Unico limite, la variabile *A$* non dovrà superare i 13 caratteri, in quanto il successivo *Chr$ (44)* (codice Ascii della virgola) e i due caratteri della ID si sommeranno alla sua lunghezza, provocando un errore *String too long* se il totale supera 16.

80 COLONNE PIXEL PER PIXEL

Vorrei sapere se, una volta entrati in Hires 640 x 200 settando il bit 7 del registro 25 del VDC, è possibile visualizzare i pixel con colori differenti.
(Raffaele Cirillo - Trecase)

Due pixel contigui, considerando il colore di primo piano, non possono assumere differenti cromatismi.

In teoria, è possibile adoperare un diverso colore per ogni raggruppamento di pixel corrispondente ad una posizione - carattere, ovvero 8 x 8 pixel, ma nella pratica è impossibile, a meno di non ridurre le dimensioni della pagina grafica.

Vediamo perchè, delineando le caratteristiche grafiche generali della modalità 80 colonne.

Com'è noto, il *Chip Vdc* dispone di una sua propria area di memoria, completamente separata dal resto del computer, per un totale di 16 Kbyte.

Quest'area, normalmente, contiene la memoria di schermo (ad ogni locazione corrisponde un carattere sul video), gli attributi di ogni carattere (colore, sottolineatura, set maiscolo / minuscolo, ecc.), nonchè i due set di caratteri disponibili e un po' di Ram libera.

Quando si passa in modo grafico, settando per l'appunto il bit 7 del registro 25 del Vdc, a partire dalla locazione 0 della stessa area di memoria si instaura una Bitmap, cui

```
100 REM  DEMO UTILIZZO DELLA SUPER HIRES 640X200
110 REM  ====================================
120 :
130 REM -------  ATTIVA GRAFICA SUPER HIRES ------
140 FAST: SCRIVI = 52684: LEGGI = 52698: SYS  LEGGI, , 25
150 RREG A: A = A AND 191: REM BIT 6 AZZERATO
160 A = A OR 128: REM  BIT 7 SETTATO
170 SYS  SCRIVI, A, 25: REM HIRES SENZA ATTRIB. COLORE
180 REM -------  PULISCE PAGINA GRAFICA  ---------
190 SYS  LEGGI, , 24: RREGA: A = A AND 127: REM BIT 7 = 0
200 SYS  SCRIVI, A, 24: REM OPERAZIONE DI 'FILL'
210 SYS SCRIVI, 0, 18: SYS SCRIVI, 0, 19: REM IND.DI START
220 SYS SCRIVI, 0, 31: REM VALORE DA STORARE IN RAM
230 FORX = 0TO63: SYS SCRIVI, 255, 30: NEXT: REM FILL
240 REM -----  RILEVAMENTO TASTI E JOYSTICK  -----
250 X = 320: Y = 100: C1 = 0: C2 = 15: REM COORDINATE E COLORI
260 GET X$: IFX$ = "S"ORX$ = "P" THEN BEGIN
270 C1 =  (C1- (X$ = "S") ) AND15: C2 =  (C2- (X$ = "P") ) AND15
280 SYS  SCRIVI, C2*16 + C1, 26: BEND: REM CAMBIO COLORI
290 IF X$ = CHR$ (19)  THEN 250: REM POSIZIONE 'HOME'
300 IF X$ = CHR$ (147)  THEN 210: REM CANCELLA SCHERMO
310 IF (JOY (2) AND128)  = 128 THEN FL = 1: REM FIRE PREMUTO?
320 J = JOY (2) AND15: IFJ = 0 THEN 260: REM RILEVA JOYSTICK
330 REM -----  NUOVE COORDINATE X ED Y  ----------
340 Y2 = Y: Y = Y +  (J < 3 OR J = 8) - (J > 3 AND J < 7)
350 IFY < 0ORY > 199 THEN Y = Y2: REM LIMITE VERTICALE
360 X2 = X: X = X- (J > 1 AND J < 5)  +  (J > 5)
370 IFX < 0ORX > 639 THEN X = X2: REM LIMITE ORIZZONTALE
380 REM ---- INDIRIZZO NELLA VDC RAM  (HI/LOW)  ----
390 IND =  (Y*80)  + INT (X/8)
400 HI = INT (IND/256) : LOW = IND- (HI*256)
410 REM --------  DISEGNA NELLA BITMAP -----------
420 SYS SCRIVI, HI, 18: SYS SCRIVI, LOW, 19
430 IFFL = 1 THEN FL = 0: GOTO260: REM SE FIRE,  NON DISEGNA
440 SYS LEGGI, , 31: RREG A: REM LEGGE REGISTRO 31
450 SYS SCRIVI, HI, 18: SYS SCRIVI, LOW, 19
460 SYS SCRIVI, A OR 2^ (7- (X AND 7) ) , 31
470 GOTO 260
```

ad ogni pixel "acceso" corrisponde un bit posto ad 1.

Tra l'altro, con una disposizione più semplice di quella adottata dal Vic per la comune Hires 40 colonne.

Qui, infatti, ad ogni linea orizzontale sullo schermo corrisponde una sequenza continua di byte.

Il colore viene associato (stavolta esattamente come per il Vic) mediante una serie di locazioni chiamate *Attribute Memory*, ognuna delle quali controlla tutti i pixel all'interno di una griglia 8 x 8 corrispondente ad una posizione carattere.

Quest'area di memoria può essere abilitata, o meno, agendo sul bit 6 del registro 25 (0 = off 1 = on); se la si attiva, l'indirizzo di partenza è determinato dal valore contenuto nei registri 20 (Alto) e 21 (basso).

A parte la complicazione in sè, nella stragrande maggioranza dei casi è conveniente disabilitare questa possibilità, per un motivo che risulterà ovvio con un paio di calcoli.

Data la risoluzione dello schermo, la bitmap occuperà 640 x 200 = 128000 bit, pari a 128000/8=16000 byte, ovvero quasi tutta la memoria del Vdc: in pratica, non rimane spazio per gli attributi colore.

La prassi più comune, quindi, è quella di disabilitare la Attribute Memory in modo tale che i colori di sfondo e primo piano, unici per tutta la pagina grafica, siano prelevati dal registro 26.

Più in particolare, i bit da 0 a 3 di quest'ultimo determinano il colore di fondo, mentre quelli da 4 a 7 il colore di primo piano.

Va detto che, volendo, gli attributi colore possono essere implementati diminuendo il numero di righe dello schermo (= limitare le dimensioni della bitmap), ma programmare la super hires è già abbastanza complesso senza aggiungere ulteriori complicazioni... Giusto per non fermarsi alle sole parole, in queste pagine potete trovare un listato dimostrativo, scritto in basic, che consente di disegnare con il joystick (in porta 2) nella pagina Hires 640 x 320. Inoltre, con il tasto *S* (da premere 2 volte, come di consueto, nelle 80 colonne) si cambia il colore di Sfondo, mentre al tasto *P* è demandato il compito di modificare il colore di tracciamento.

Premendo il *Fire* del Joy, si avanza (un po' alla cieca) senza disegnare, con *Home* ci si riporta al centro dello schermo e con *Shift + Home* si cancella tutta la pagina.

Così com'è, il programma non prevede un ritorno alle normali condizioni testo, che richiederebbe quantomeno un ripristino della ram caratteri; per farlo, è necessario un *Reset* del sistema.

Il listato, zeppo di Rem esplicative (eliminabili), ha uno scopo prettamente didattico e dimostrativo, ma può servire da base per più sofisticate elaborazioni. L'ideale, comunque, sarebbe tradurlo in Assembly per supplire alla lentezza del basic, peraltro più comprensibile (o quasi...).

Si ricordi, spulciando il listato, che per accedere ai registri del *Vdc* è necessario ricorrere a due routine di sistema: *$CDCC* per pokarvi qualcosa e *$CDDA* per leggerli.

Nel primo caso, in basic, si adopera...

Sys 52684, valore, registro

...mentre con...

Sys 52698, , registro

...si otterrà il contenuto del registro nell'accumulatore, che da basic è associabile alla variabile (p. es.) A mediante l'istruzione *Rreg A*. Dei registri Vdc usati, 18 e 19 servono per memorizzarvi l'indirizzo della Ram 80 colonne al quale si vuole accedere (formato alto / basso); il bit 7 del registro 24 indica se si vuole compiere una operazione di *Fill* (bit azzerato, come nel nostro listato per porre a zero tutti i bit della bitmap) o di *Copy* (bit settato); infine, il registro 31 deve contenere il valore che si intende trasferire nella Ram puntata dai registri 18 e 19 (tanto per il Riempimento che per il semplice "storaggio"). Quando si scrive qualcosa in Ram (nel nostro caso rasformata in Bitmap), il contenuto dei registri 18 e 19 viene automaticamente incrementato dal sistema. Il tutto non è certo semplice, ma la complicazione, per chi vuole "strizzare" il C/128, è prassi.

GIA' FATTO, O QUASI...

Ho trasferito sul C/128 una routine basic di Input controllato in origine scritta per il C/64, sul quale funzionava piuttosto bene. In modo 128, però, pur non presentando anomalie, è molto più lenta: non potreste pubblicarne un equivalente in linguaggio macchina tanto per il C/64 che per il C/128?

(Renato Scuccato - Carmignano di Brenta)

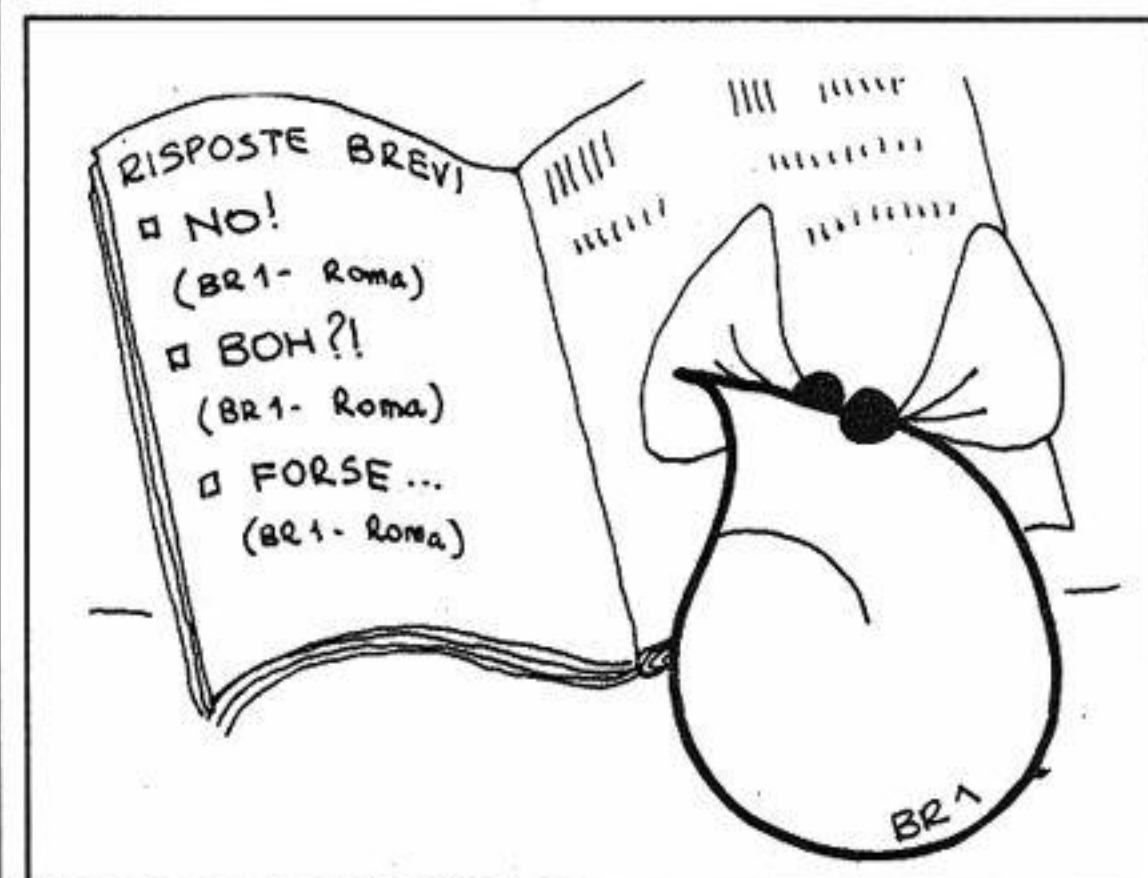
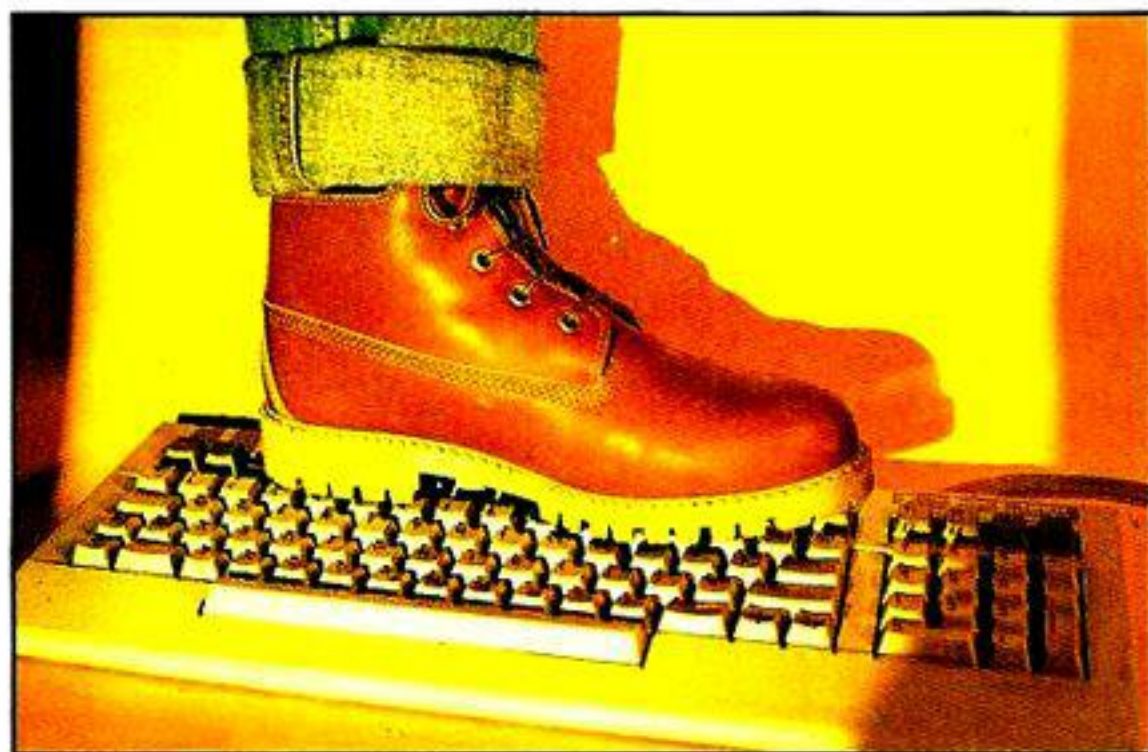
Il basic 7.0, in effetti, risulta spesso più lento del suo meno potente cugino 2.0, a meno di non sfruttare appieno le possibilità offerte dal *Fast Mode* (su 80 colonne è davvero risolutivo, potendolo adoperare in permanenza).

Nel caso particolare degli Input controllati, spesso questo tipo di routines sono progettate in modo da potersi adattare alle più svariate esigenze, con inevitabile maggiore "pesantezza" nel procedere.

Un primo rimedio, già in basic, può dunque essere quello di snellire la routine mantendo solo i controlli strettamente necessari al programma.

Chiaro che, affidando il tutto al linguaggio macchina, il problema cessa di esistere.

Dovendo, però, progettare una routine necessariamente generica, questa assumerebbe in ogni caso delle dimensioni non certo irrilevanti, senza peraltro potersi *mai* adattare a tutte le esigenze.

Una routine (in l.m.) del genere, comunque, è stata già pubblicata su C.C.C. n. 55 (Lug / Ago '88), riferita al C/64: con poche modifiche, e un po' di abitudine all'Assembly, può anche essere adattata al modo 128.

IL C/128 VA IN PENSIONE

Ed il C64 entra nella terza età.
Consigli (quasi) disinteressati per guardare lontano

di Alessandro de Simone

Dopo la morte ormai accertata del *C/16* e del *Plus/4* (annunziata con rammarico, del resto, qualche numero fa) il polso del *C/128* incomincia a battere debolmente.

La mortale malattia è quella che, puntualmente, ha colpito dapprima i sistemi della serie Commodore *Pet* e poi il *Vic 20*.

Ma il "virus", stavolta, risulta ancora più violento di quanto accadde, tempo fa, ai primi modelli di casa Commodore.

Alla serie Pet ed al Vic 20, infatti, l'obsolescenza inferse il colpo mortale per motivi tecnologici: i primi erano computer in grado di gestire al massimo 32 KRam ed offrivano l'uscita solo in bianco e nero; il secondo, pur disponendo di un'uscita a colori, era troppo limitato sia per quantità di Ram sia per l'angusta visualizzazione dello schermo (512 caratteri o giù di lì). Era fin troppo intuitivo che il *C/64*, appena uscito, dovesse far cadere nell'oblio tutti i computer che, a pari prezzo, offrivano caratteristiche analoghe, se non inferiori.

Il C/64, ed in seguito il *C/128*, offrivano infatti, a parità di prezzo, una quantità di memoria decisamente all'avanguardia, un'uscita a 16 colori (solo più tardi l'IBM gridò al miracolo quando propose un Personal Computer a quattro colori) e la gestione innovativa di musica e sprite.

Fino all'anno scorso, dunque, è durata la pacchia; che, a nostro parere, è durata fin troppo.

Oggi un C/64 dotato di drive *1541* costa, all'utente finale, quasi quanto un *Amiga 500*. E' ben vero che un Amiga senza secondo drive, nè espansione di memoria, nè monitor dedicato vale ben poco.

Tuttavia, volendo considerare un sistema completo di *stampante* e di qualche ammenicolo pressochè indispensabile, oggi come oggi il C/64 (ed a maggior ragione un C/128, peraltro non più presente nei listini ufficiali della casa madre) non può assolutamente reggere il passo con ciò che offre la "concorrenza", sia questa la stessa Commodore (Amiga e sistemi Ms-Dos compatibili), siano altre marche (Atari, Ms-Dos di vario tipo).

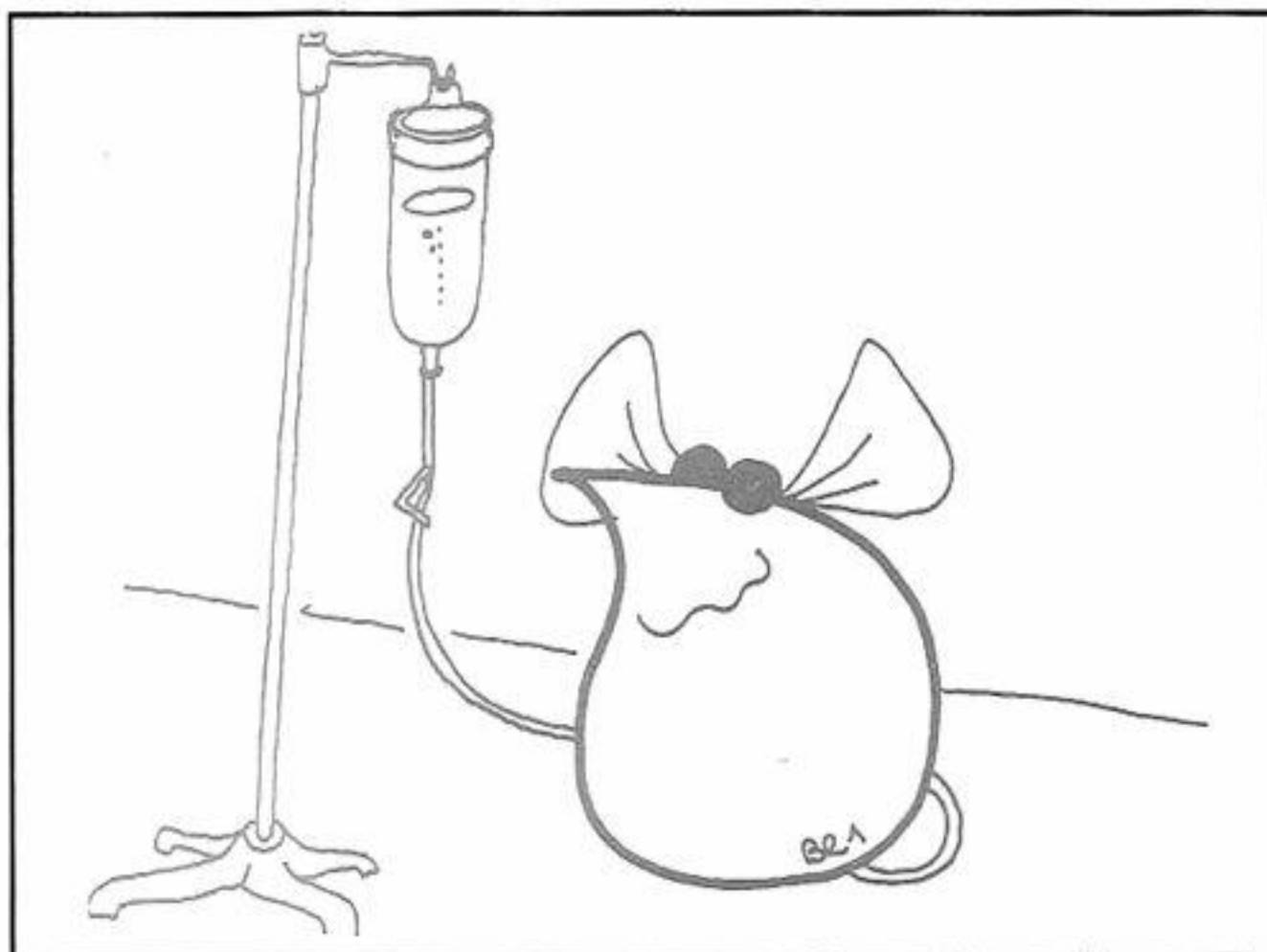
Il prezzo del sistema in possesso del nostro lettore è senz'altro pari (se non superiore) a quello di un compatibile Ms-Dos e può aver senso solo se si considera che qualche anno fa, al momento dell'acquisto, un clone IBM costava almeno il doppio.

SOLDI, E NON SOLO

La morte del C/128, quindi, è dovuta non solo ad obsolescenza *tecnologica*, ma anche, e soprattutto, a quella *commerciale*. Nessuno, oggi, spende un milione per entrare in posseso di un C/128 (o C/64) completo: preferisce acquistare un *clone* e, un po' più in là, la stampante per completare il sistema. L'unico motivo che spinge il neoutente ad acquistare un C/64 è il fatto che questo è venduto, quasi sempre, con il registratore a cassette in omaggio (o quasi) e che si può fare qualcosina anche utilizzando quest'unica periferica, soprattutto nel campo dei videogames. Spingere l'utente (che già possiede il solo C/64) ad acquistare, oggi, anche il drive, può risultare *antistorico*, a meno che la Commodore non decida di vendere i suoi 1541 ad un prezzo non superiore alle 200 mila lire (IVA compresa, per carità). Ma, in questo caso, si potrebbero (forse) limitare gli effetti dell'obsolescenza solo nei *vecchi* sistemi: chi non possiede un computer, e desidera, finalmente, acquistarne uno, farà molto bene i suoi conti prima di spendere più di mezzo milione per un C/64 completo di drive.

SOFTWARE

Ma anche ammesso, e non concesso, che l'utente sia perdutamente innamorato dei videogames del C/64 (notate che non parliamo già più del C/128?) e degli effetti sonori impossibili da realizzare con un Ms-Dos compatibile, un'altra limitazione provvede ad allargare il già ampio divario tra il computer ad 8 bit e quelli, più moderni, a 16/32 bit.

Il software, ragazzi, non è una cosa che si può trascurare. E' ben vero che esistono pacchetti di tutto rispetto che consentono di archiviare dati, stendere relazioni e documenti (perfino libri!) ed effettuare calcoli in catena. Tutto sta a restare delle stesse convinzioni anche dopo aver usato, per un solo pomeriggio, pacchetti analoghi su *veri* Personal Computer.

La velocità di elaborazione, le raffinatezze tecnologiche, l'affidabilità e, soprattutto, le *potenzialità* offerte da un *qualsiasi* pacchetto professionale che giri su un *qualsiasi* personal computer mettono in secondo piano *tutte* le altre caratteristiche del C/64, ad eccezione dei motivi affettivi.

"Fare" *D.T.P.* con C/128 e stampante Mps-803? Certo, l'ho fatto anch'io; ma solo quando di D.T.P. non si parlava nemmeno e l'unico programma del ramo alfa-numerico-pseudo-grafico (*Print Master*, insomma) girava timidamente sui P.C. quando il C/64 già offriva, da tempo, *Geos, Print Master, Print Shop, News Room, Music Shop* ed altri ancora.

Paragonare, oggi, le caratteristiche di Geos 64-128 con *Ventura Publisher, Pagemaker* o *Pagesetter* è una vera follia, scusabile solo in chi non li ha *mai* visti girare.

AMO IL C/128 PERCHE'...

Premetto che sono un felice possessore di un sistema composto da C/128, drive 1541, drive 1571, stampante Mps 803, monitor 1901, mouse 1351, interfaccia seriale Rs-232, Modem 300/1200 baud Hayes compatibile.

Mi rivolgo a tutti gli utenti e non (leggi negozianti) di sistemi Ms-Dos, Amiga, Atari (ed altro) i quali denigrano fin troppo questa stupenda macchina, relegandola al ruolo di "videogame casalingo".

Personalmente ritengo che non sia da sottovalutare, anche se la carenza di software dedicato la mette in secondo piano; non bisogna dimenticare che dal punto di vista h/w è abbastanza completa: uscita 40 / 80 colonne, Rs-232, bus seriale veloce, grafica da 160 x 200 a 640 x 200 e così via.

Forse il mio è un caso isolato, almeno da quanto leggo su riviste del settore, ma sono riuscito a procurarmi quasi il meglio di quanto si possa oggi trovare sul mercato in fatto di s/w in modalità 128, e per giunta in 80 colonne: *Superbase, Superscript* nonchè l'eccezionale *Geos 128 2.0*, tutti rigorosamente originali (sono contro ogni tipo di pirateria informatica); inoltre posseggo un buon compilatore Basic 7.0 *(Gnome Speed)* ed un programma di comunicazione *(Link 128)*.

A conti fatti, quindi, non vedo questa superiorità schiacciante, come almeno vogliono far credere, del mondo Ms-Dos nei confronti del mio C/128, visto che comunque sono in grado di archiviare dati e richiamarli ad una discreta velocità, redigere documenti di lunghezza superiore alle capacità di un dischetto da 340 Kbytes, fare del Desk Top Publishing *casereccio*, comunicare su rete telefonica e così via.

In conclusione esprimo il mio desiderio informatico: Apple McIntosh II, che è l'unica macchina per la quale rinuncerei al mio attuale sistema.

(Antonio Rotta - Arco Felice)

CONCLUDENDO

Solo chi già possiede il C/64-128 (e non ha voglia di spendere altri soldi nel settore informatico) può accontentarsi di rinviare (perchè di questo solo si tratta) l'acquisto di un nuovo sistema.

Tutti gli altri, che inizino o sieno a metà dell'opra, si sbrighino a decidere.

L'obsolescenza è un fenomeno inarrestabile che sta colpendo mortalmente i nostri amati 64-128, giorno dopo giorno (date un'occhiata agli annunci economici se non ci credete).

Noi di *Commodore Computer Club* abbiamo infatti deciso di parlare ancora del C/128 solo fino al gennaio del 1991 e solo se le richieste dei nostri lettori saranno sufficientemente convincenti.

Del C/64 dovremmo (il condizionale è d'obbligo) parlare ancora fino al *dicembre del 1991*, e sempre a patto che i lettori, vecchi e nuovi, ce lo consentano.

Le pagine lasciate libere dalle vecchie tastiere saranno, a mano a mano, sostituite da argomenti relativi ad **Amiga** e (se necessario) ai sistemi **Ms-Dos**.

CAMPUS

64 / 128

18 - IL C/64 SI TRASFORMA IN OROLOGIO - SVEGLIA

Il piccolo computer della Commodore (ed il suo fratello "maggiore", il C/128) dispone di due circuiti integrati, non molto conosciuti dalla stragrande maggioranza degli utenti. In tali dispositivi sono incluse alcune funzioni decisamente interessanti; ci riferiamo all'orologio, capace di tener conto perfino dei decimi di secondi, ed alla sveglia ad esso collegata. Inutile dire che tale contatempo funziona per proprio conto e, per farlo "venire alla luce", occorre attivare una routine in Interrupt che esamini, ogni sessantesimo di secondo, i registri del dispositivo. Per rendere l'elaborazione interessante e completa viene presentata una procedura davvero complessa, ma relativamente semplice da digitare, in grado di trattare, contemporaneamente, raster, interrupt, sprite ed attivazione dell'orologio sveglia. Un programma di sicuro interesse, almeno per i lettori più in gamba.

29 - HARD COPY PER C/128

Riportare su carta la pagina grafica è una delle funzioni più richieste dall'utenza del C/128. Di solito, però, bisogna ricorrere a routines in linguaggio macchina che, a parte la "seccatura" della difficoltà di digitazione, possono occupare aree di memoria non sempre libere o determinabili a priori. Viene presentata una procedura che, ricorrendo esclusivamente ad istruzioni Basic, consente di riportare su carta una qualsiasi pagina grafica del C/128.

Nessun albero viene abbattuto per gli inserti di *Commodore Computer Club*, stampati su carta riciclata al 100%

LE AVVENTURE DI

PRIMO GIOVEDINI

by M. Mietta B. DeToffoli

Gara di acrobazia (3° e 4° file)

IL C/64 SI TRASFORMA IN OROLOGIO - SVEGLIA

C'è di tutto: sprite oltre il bordo, interrupt, raster e perfino l'utilizzo di un circuito elettronico poco noto

di Paolo Monti ed Edmondo Fichera

Il C/64 ed il C/128 dispongono di un paio di circuiti elettronici molto sofisticati

A molti utilizzatori sarà senz'altro capitato di avere un'idea improvvisa per un programma e di mettersi subito alla tastiera, senza però tenere conto di un elemento piuttosto importante: *il tempo.*

Chi si accinge ad editare un programma non sempre ha l'intero pomeriggio a disposizione, magari a causa di altri impegni, ed è quindi costretto a guardare in continuazione l'orologio, perdendo così il "filo" e la concentrazione.

Alle volte può anche capitare che, essendo il programma particolarmente interessante, il programmatore si lasci prendere dall'enfasi della digitazione e perda completamente la cognizione del tempo, dimenticandosi addirittura di avere qualche importante appuntamento.

E' così che abbiamo deciso di venire incontro alle esigenze di coloro che, pur essendo sommersi da altre occupazioni, non esitano a dedicare parte del loro prezioso tempo al computer.

Il programma di querste pagine, infatti, visualizza sul bordo inferiore dello schermo un preciso orologio dotato di sveglia programmabile a piacere. I vantaggi che ne derivano sono numerosi.

- non è necessario distogliere lo sguardo da ciò che si sta digitando;
- grazie alla sveglia programmabile risulta "impossibile" saltare gli appuntamenti;
- essendo situato sul bordo, e non nell'area video, l'orologio non sottrae nemmeno una linea allo schermo;

Inoltre è intuibile che il programma si rivela interessante anche dal punto di vista didattico, oltre che pratico, visto che utilizza tecniche di *interrupt* e di *raster*, gestisce le *CIA* e gli *sprites* nonchè di alcuni fondamentali *registri* del Vic II.

I due circuiti di adattamento delle interfacce del C/64 (le CIA) oltre a gestirne le linee in ingresso e in uscita, possiedono un orologio completo di ore, minuti, secondi e, addirittura, *decimi* di secondo.

Parametro	CIA 1	CIA 2
Decimi	56328	56584
Secondi	56329	56585
Minuti	56330	56586
Ore e a.m. / p.m.	56331	56587

Parametri e locazioni delle CIA nel C/64

I parametri dell'orologio sono distribuiti in quattro registri consecutivi che, per le rispettive CIA, corrispondono alle locazioni indicate in tabella.

Nel nostro programma verrà usato il primo dispositivo.

COPIALO PER TELEFONO

Anche i listati presenti in queste pagine possono esser tirati giù per mezzo del modem; se, ovviamente, ne possedete uno.
La procedura per collegarsi con la nostra banca dati (attiva 24 ore su 24) è riportata su altra parte della rivista.
Chi non possiede il modem (che aspettate a procurarvelo?) può tuttavia richiedere il dischetto, contenente il software, presso il nostro servizio arretrati.

Per impostare i parametri dell'orario è sufficiente trascrivere i valori desiderati nell'opportuno registro, tenendo però conto di alcuni elementi: innanzi tutto si deve sapere che se si effettua una scrittura nel registro delle ore, l'orologio si bloccherà istantaneamente e ripartirà solo al momento di una scrittura nel registro dei decimi di secondo, mentre se si scrive prima in un altro registro verrà eseguita la scrittura, senza l'arresto del dispositivo.

L'utente ha così la possibilità di settare a piacimento l'orario, e di farlo partire all'istante desiderato, se avrà l'accortezza di modificare i registri *partendo* dall'ora e *finendo* con i decimi di secondo.

Anche in lettura si ha una situazione analoga; se, infatti, leggiamo il registro dell'ora, verranno simultaneamente congelati gli altri tre registri nel cosiddetto *circuito Latch.* In pratica questo circuito fa sì che le quattro locazioni di memoria sopra menzionate rispecchino costantemente il contenuto dei registri all'istante della lettura delle ore, mentre in realtà l'orologio continua a contare normalmente. Con ciò l'utente può leggere con tutta calma i vari registri fino a che, con una lettura dei decimi di secondo, il Latch verrà sbloccato rispecchiando così i valori correnti dell'orario.

Se viene letto un qualsiasi altro registro *prima* di quello delle ore, esso viene letto al volo, cioè presentando il valore istantaneo del parametro scelto (minuti, secondi oppure decimi).

Ovviamente, per leggere e scrivere nei registri, si usano le istruzioni Basic *Peek* e *Poke* o quelle assembly *Lda* e *Sta.*

Passiamo ora a esaminare *come,* cioè in quale formato vanno trascritti i parametri nei registri.

DALLA TEORIA ALLA PRATICA

Le quattro locazioni considerate hanno una particolarità: non funzionano come tutte le altre, ma vanno considerate come *spezzate in due,* cioè nei quattro bits alti e nei quattro bassi; i primi rappresentano le *decine* del numero scritto, i secondi le *unità.*

BIT	7654	3210	
	0011	0101	= 35

Esempio di settaggio dell'orologio

Infatti, come vediamo nello schema, i bits più alti contengono il numero binario 3, mentre i più

VIETATO AI MINORI

E' probabile che i neo-utenti di computer non riescano a ben comprendere l'utilità della procedura software descritta nelle presenti pagine. Questa è infatti destinata a coloro che sono già padroni delle principali tecniche di programmazione e desiderano approfondire il modo in cui un elaboratore organizza i vari dati all'interno della memoria.
Il programma, pertanto, rappresenta un invito ai lettori più esperti, soprattutto a coloro che, considerando il listato come una base di partenza, riescano a pervenire a procedure più interessanti e capaci, magari, di offrire applicazioni di più ampio respiro.
Se hai incominciato da poco, comunque, non scoraggiarti! Tutti coloro che, oggi, vantano una particolare competenza nel campo dell'informatica hanno iniziato con un banale Print "Pippo". Perchè non dovresti riuscire anche tu?...

Per settare correttamente l'orologio e la sveglia è necessario attenersi alla procedura descritta per evitare "regolazioni" imprecise

Grazie, va molto meglio... Stavo dicendo che oggi vedremo la fine della storia della gara...

Il settimo bit della locazione 56331 è incaricato di tener conto del formato AM/PM

bassi il numero 5. Quindi, per scrivere il numero correttamente, *non* dobbiamo scrivere (per i minuti, ad esempio)...

Poke 56330, 35

...bensì...

*Poke 56330, 3 * 16 + 5*

...poichè solo così il *3* verrà scritto nella metà (in gergo: *nibble*) alta del byte e il *5* in quella bassa.

La locazione delle ore presenta una piccola differenza: in essa, infatti, il bit 7 fornisce l'indicazione AM / PM; se il bit è a uno, l'ora è da considerarsi *pomeridiana* (dopo mezzogiorno); viceversa è antimeridiana (prima di mezzogiorno).

Esempio:

Bit	7654	3210
	1001	0001

Poke 56331, 128 + (1 * 16) +1

Con ciò avete inserito, nel registro, *una* decina ed *una* unità, che insieme indicano le ore *undici*; avete inoltre settato l'indicatore su PM, quindi si intenderanno le undici di sera.

Perchè il dispositivo funzioni correttamente c'è da tener presente un'ultima cosa: il dispositivo necessita, per problemi di sincronismo, della frequenza della corrente di rete che, in Italia, è di 50 Hertz; dal momento che la CIA è settata per default sui 60 Hertz (paesi angloamericani) bisogna settare a *uno* l'apposito bit della locazione *56334* (*56590* per CIA 2), e per la precisione il bit 7, mediante una semplice...

Poke 56334, Peek (56334) Or 128

Se lasciassimo il bit a zero, accadrebbe che... un secondo *non* durerebbe *un* secondo, ma molto di più!

ALLARME!

Come accennato inizialmente, il programma fa uso di un'altra caratteristica del dispositivo TOD (*Time Of Day* = tempo del giorno = il nostro orologio!) e cioè dell'*allarme* programmabile.

Infatti il registro 15 della CIA 1, allocato in 56335, (56591 per CIA 2), contiene un bit, l'ottavo, che, se impostato, permette di regolare l'allarme agendo sulle medesime locazioni prima esaminate per l'orario. Anche il formato con cui inserire il parametro resta immutato, come anche la possibilità di determinare se si tratti di ora pomeridiana o antimeridiana mediante il bit 7 del registro delle ore.

Un esempio:

Supponiamo di voler programmare la sveglia alle 7 e 30 del mattino.

```
Poke 56335, Peek (56335) Or 128 : Rem imposta modo sveglia
Poke 56331, 7: Rem AM + 0 decine e 7 unità
Poke 56330, 3 * 16 + 0: Rem 3 decine 0 unità
Poke 56329, 0: Rem 0 secondi
Poke 56328, 0: Rem 0 decimi di secondo
```

Una volta settata la sveglia, si riporta a zero il bit, prima impostato, con...

Poke 56335, Peek (56335) And 127

...per riavere il controllo dell'orario; nell'istante in cui l'orario eguaglia il valore inserito nell'allarme, il dispositivo imposta ad uno il terzo bit della locazione 56333 (56589 per CIA 2) per avvisare l'utente, il quale agirà di conseguenza; è possibile far sì che il dispositivo generi anche una richiesta di interruzione, oltre che settare il bit appena esaminato, permettendo, magari, l'esecuzione automatica di una routine appositamente scritta.

Per ottenere questo è sufficiente abilitare l'interruzione d'allarme scrivendo a *uno* contemporaneamente il bit 7 e il bit 2 della 56333 mediante l'istruzione...

Poke 56333, Peek (56333) Or 132

La locazione 56333 contiene infatti un registro vitale della CIA1 e cioè il registro di controllo degli interrupt provenienti dai vari dispositivi dell'interfaccia; vi sono cinque diverse sorgenti di interruzione nelle CIA e precisamente:

1) timer a

2) timer b
3) allarme
4) porta seriale
5) flag

Esse sono disposte nel registro come nella tabella riportata in queste pagine.

Bit	Contenuto
0	Timer a
1	Timer b
2	Allarme
3	Ps
4	Flg
5	non usato
6	non usato
7	Imposta / Cancella

Quando scriviamo in questo registro, abbiamo la possibilità di *mascherare*, o meno, una qualunque di queste sorgenti, facendo sì che, al momento della richiesta di interruzione, questa possa giungere fino al microprocessore (che si "interromperà") o venga bloccata.

Per abilitare l'interruzione bisogna, come avrete intuito, settare a *uno* contemporaneamente sia il bit della sorgente prescelta, sia il bit 7; se scriviamo a *uno* solo il bit della sorgente, essa verrà mascherata e non provocherà interruzioni della CPU.

In lettura, in qualunque caso, la sorgente di interruzione, al momento della richiesta, setterà *almeno* il proprio bit della locazione 56333, cosicchè basta leggere costantemente il registro e controllare quale sorgente chiede l'interruzione.

Nel nostro caso, se controlliamo il bit 2, ci accorgiamo di sicuro se è scattato l'allarme.

IL DEMO IN BASIC

Il nostro programma Basic, unito alla routine in linguaggio macchina, provvede all'esecuzione di tutte le regolazioni finora esaminate; l'utente dovrà solo inserire l'ora corrente e la sveglia, da digitare nel formato *HHMMSS* cioè usando due cifre per ciascun parametro senza separazione tra l'uno e l'altro; inoltre le ore vanno da *00* a *23*, tipico dei paesi europei, a differenza di quelli angloamericani che presentano la differenza tra *AM* e *PM*.

Per esempio, se alla richiesta della sveglia digitassimo *033000*, ci sveglieremmo nel cuore della notte!

Il programma visualizza, sul bordo inferiore dello schermo, l'orologio che indica l'orario appena settato e chiede di premere *S* per farlo partire; una volta azionato, non ci rimane che digitare *New* e iniziare serenamente la digitazione del nostro nuovo programma, certi che, al momento giusto, il nostro fedele C/64 ci avviserà che, ahimè, il tempo è scaduto.

LA ROUTINE L. M.

Passiamo ora ad una descrizione più tecnica e dettagliata del programma Basic e della routine l.m. La tecnica per far *sparire* il bordo inferiore e superiore è stata già ampiamente esposta dal *Lorenzo Emilitri* su un altro numero della rivista e ci sembra quindi superfluo dilungarci su essa: basti ricordare che tale tecnica si basa sul fatto che si *inganna* il Vic facendogli credere di aver già disegnato il bordo.

Inoltre è stato necessario spostare l'editor di schermo nel banco *Vic 3* (posizionato da 49152 a 65535), visto che il raster riempirebbe la parte di bordo ora resa visibile con il contenuto dell'ultima locazione del banco attuale del Vic; questa locazione, normalmente, è la 16383, cioè nel bel mezzo della Ram Basic, e per ragioni estetiche conviene che contenga zero in modo che non compaiano delle bande nere. Così facendo, però, i programmi Basic dovrebbero terminare ben prima della 16383, rendendo inutilizzabile oltre metà della Ram disponibile. Cambiando il banco del Vic dallo *0* al *3*,

La locazione 56334 è di vitale importanza per la "scansione" del tempo

La storia riprende. Avevamo lasciato il caro Iceman in seria difficoltà: durante un volo di allenamento egli aveva voluto strafare, ritrovandosi così su un aereo privo di controllo in un territorio ignoto, ben al di là dei 64 KRam normalmente conosciuti...

La manipolazione del Raster consente di visualizzare gli sprite sul bordo del video

l'ultima locazione del banco diviene la 65535 e azzerarla non comporta alcuna conseguenza.

I dati per i "disegni" delle dieci cifre necessarie per visualizzare l'orologio (da 0 a 9) e del carattere separatore del doppio punto (:) vengono prelevati direttamente dalla Rom dei caratteri e riversati in una zona Ram a cui, poi, indirizzeranno i puntatori agli sprites che ora si trovano a partire da 49152 + 2040 a 49152 + 2047. Le righe del programma Basic che compiono tale operazione sono quelle da 20 a 40: dapprima viene azzerata una zona di memoria che possa contenere undici blocchi da 64 bytes ciascuno, poi gli undici blocchi vengono riempiti, ognuno, con solo 8 bytes corrispondenti ai dati di ciascun carattere, che gli sprites mostreranno in seguito, per di più ingranditi.

Il programma Basic, inoltre, imposta i parametri del SID e controlla che non vengano inseriti dati errati, come ad esempio la digitazione di 76 alla richiesta dei minuti.

La routine in l.m. è divisa ovviamente in due parti, la seconda delle quali gira sotto interrupt.

La prima parte provvede a spostare i puntatori degli interrupt, ad abilitare le interruzioni di raster, a ingrandire e posizionare gli sprites, a copiare i caratteri nel banco 3 del Vic e, infine, ad inizializzare alcune locazioni.

La seconda parte si occupa di far scomparire il bordo, di controllare l'interruzione d'allarme, di consultare i quattro registri dell'orario (e porne i valori letti nei puntatori agli sprites), e di saltare all'inevitabile $EA31.

Vi è infine una subroutine che viene chiamata al momento dell'allarme e fa sì che venga prodotto un tipico BIP - BIP che dura undici secondi (dal secondo *0* al secondo *10*). Questa parte funziona in modo molto semplice: la routine di interrupt viene chiamata dal raster ad ogni schermata (e quindi con una frequenza di circa 50 Hertz). Se è scattato l'allarme, anche la subroutine del BIP - BIP viene chiamata 50 volte ogni secondo; noi non abbiamo fatto altro che incrementare una locazione, ad ogni chiamata, sapendo solo che, quando fosse arrivata al valore 50 voleva dire che era passato un secondo; inoltre abbiamo ulteriormente suddiviso i cinquanta cinquantesimi di secondo in modo da produrre un doppio BIP accendendo o spegnendo il suono agli opportuni valori dei cinquantesimi di secondo.

Un'altra locazione controlla che ciò avvenga solo per undici secondi. Analizzare il disassemblato aiuterà a capire meglio il funzionamento della subroutine.

Ed ecco qui che spunta fuori una curiosità che riteniamo siano ben pochi a conoscere, compresi anche i più esperti programmatori in l.m.

Se si pone la sveglia ad un qualsiasi orario, ma con i secondi a *00*, l'allarme suonerà *anche* per il minuto *successivo* a quello stabilito.

Questo dipende dal fatto che quando i secondi del minuto in cui è suonato l'allarme passano da 59 a 00, il dispositivo TOD impiega qualche frazione di secondo per incrementare di uno anche i minuti e, per tale durata, l'orario risulta identico a quello del minuto precedente; il registro di interruzione della CIA, allora, ri-resetta a *uno* il bit dell'allarme e il nostro programma riesegue la subroutine del BIP - BIP.

Questo, però, non è affatto un inconveniente, anzi è un buon rimedio per ricordare (a chi si fosse soffermato sulla tastiera per concludere "quell'ultima riga di programma") che è proprio ora di smettere.

Comunque, se questo fatto invece creasse dei fastidi, è ovviamente possibile far sì che l'allarme suoni una sola volta.

E' infatti sufficiente porre la sveglia con i *decimi di secondo* non a zero, ma a *4* oppure a *5*, per dar tempo al TOD di incrementare, in tale lasso di tempo, anche i minuti; oppure, nel caso del nostro programma, basta settare la sveglia con i *secondi* diversi da zero, ad esempio a uno: in questo caso l'allarme suonerà una volta sola, ma un secondo dopo il minuto preciso, cosa questa di insignificante rilevanza.

Per concludere dobbiamo confessare una piccola pecca del programma in linguaggio macchina: esso infatti, nella parte che gira *sotto* interrupt, risulta di qualche ciclo più lungo del normale.

Questo fa sì che ogni tanto, nel momento in cui si impartisce la *Sys* di partenza, il programma non parta sincronizzato e dopo qualche millesimo di secondo si blocchi.

Nonostante le probabilità che il programma non parta al primo colpo siano decisamente piccole, (circa 1 volta su 15), ci è parso comunque doveroso informarne i lettori, i quali non hanno nulla di che preoccuparsi, visto che l'inchiodamento non è mai definitivo.

Basta infatti eseguire le seguenti operazioni per riavere il controllo del cursore:

1) battere Run / Stop + Restore

A questo punto si dovrebbe notare un cambiamento di ciò che è visibile sullo schermo.

2) digitare *ALLA CIECA*: Poke 648, 4 e battere il tasto Return.

Compariranno la scritta *Ready* ed il cursore; a questo punto potete ridare il *Run* e, a meno che non siate la sfortuna fatta persona, dopo aver reinserito ora e sveglia, comparirà finalmente il vostro stupendo orologio digitale.

A nostro parere lo spiacevole inconveniente è dovuto alla eccessiva lunghezza (che non ci è stato possibile ridurre, pena la perdita di notevoli prestazioni del programma) della routine di interrupt; ma non essendone certi, se qualcuno scoprisse che il motivo è un altro (e trovasse un rimedio efficace), potrebbe sempre comunicarlo ai lettori (noi compresi), mediante la rubrica della posta di C.C.C. o in un apposito articolo.Quello che (speriamo) verrà scoperto sarà comunque, a nostro parere, di notevole valore didattico e insegnerà qualcosa anche ai programmatori più esperti.

Non bisogna dimenticare che, al termine di una routine che altera l'interrupt, l'ultima istruzione deve essere un "salto" a $EA31

PER I PIU' BRAVI

Lavorando con l'Interrupt può capitare di scrivere routine molto lunghe e complesse.

In questi casi sorge immediatamente una difficoltà, legata direttamente alla stessa tecnica che figura alla base della gestione dell'Interrupt.

In pratica, ogni sessantesimo di secondo il computer interrompe qualsiasi operazione e "salta" ad eseguire una particolare routine (posta, nel caso del C/64, a partire da $EA31) che sovrintende a particolari funzioni, vitali per l'elaboratore.

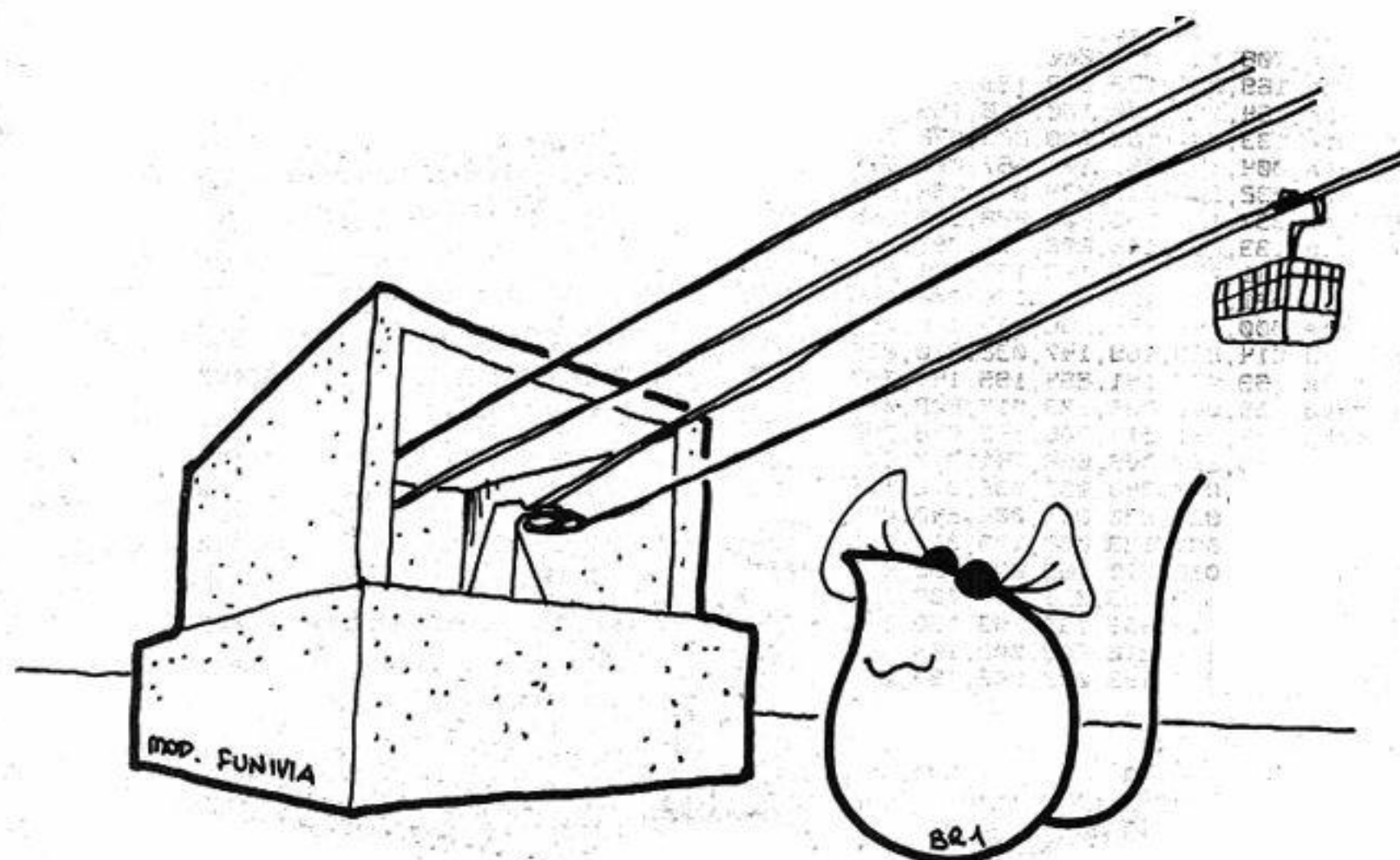
Dal momento che tale routine necessita di un certo tempo per essere eseguita, è evidente che questa "deve" essere sufficientemente breve per fare in modo che il calcolatore elabori anche il programma dell'utente.

In altre parole, supponiamo di suddividere la durata di un sessantesimo di secondo in dieci parti eguali. Se la routine di Interrupt presenta una durata di tre parti, vuol dire che il computer dedica, in definitiva, il 70% del tempo per elaborare il programma dell'utente ed il 30% le "proprie" routines.

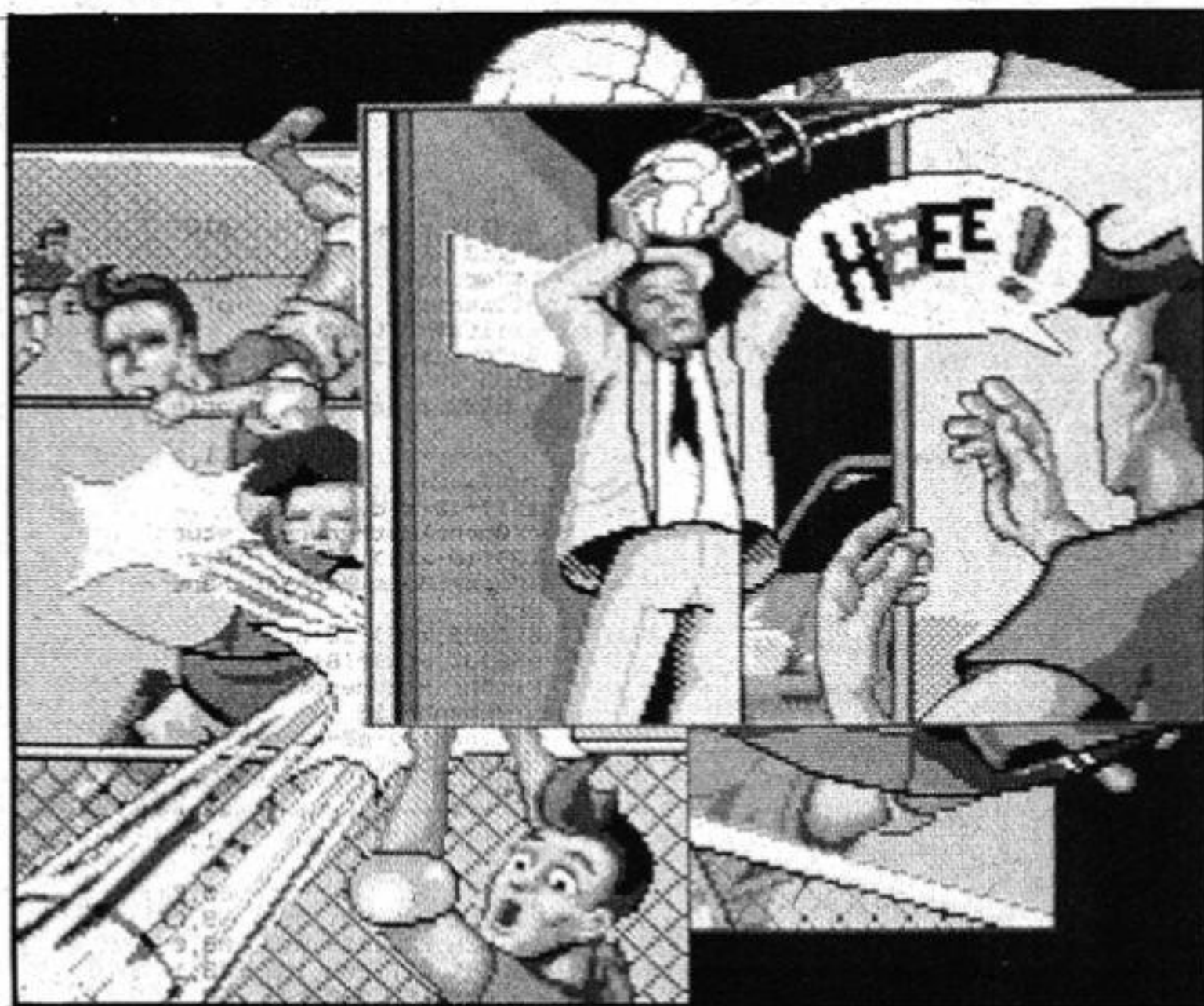
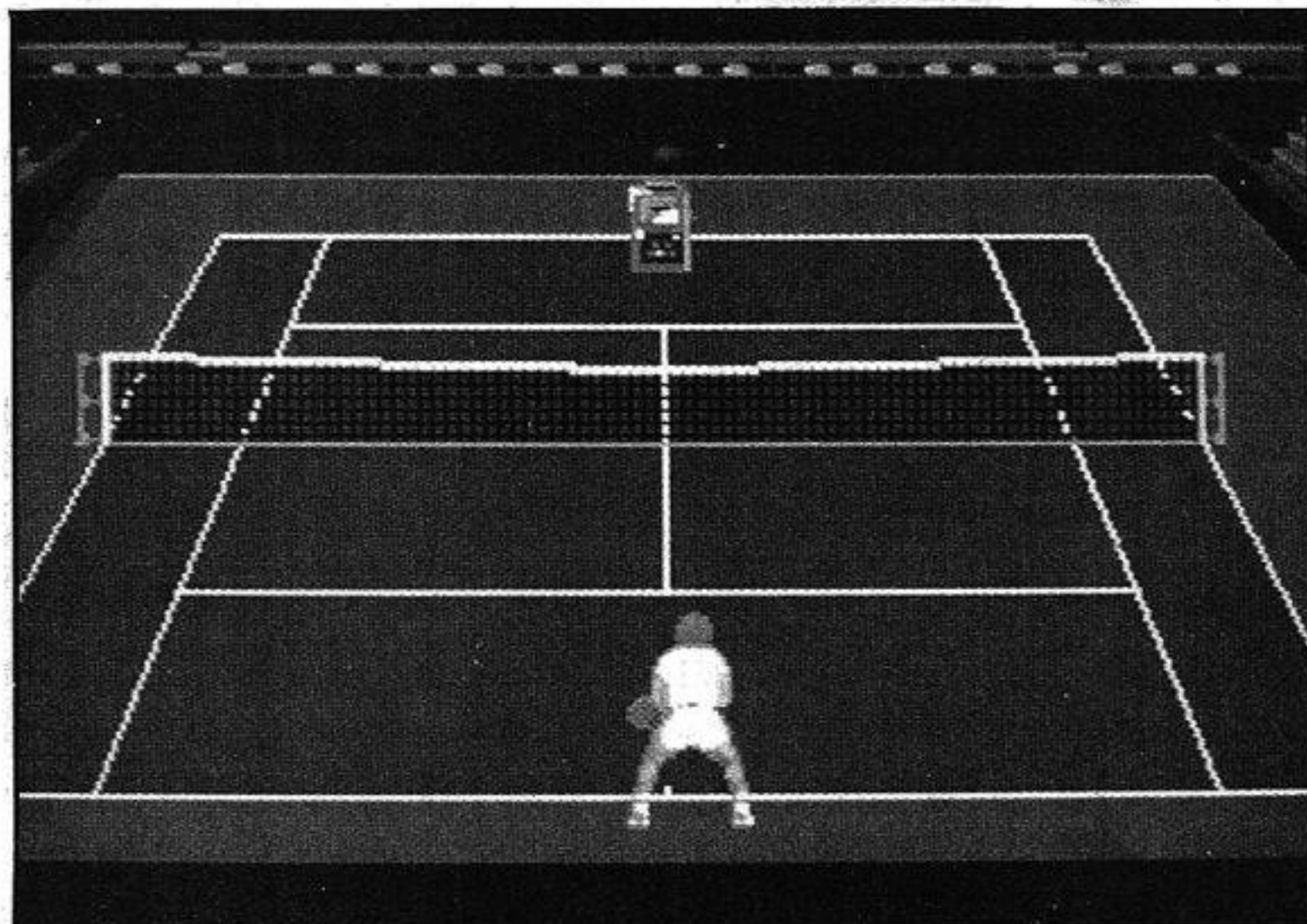
Se, invece, l'Interrupt richiede 4 parti, il tempo di elaborazione dell'utente scende al 60%. Al limite, con una durata di Interrupt di 10 parti, non vi sarà tempo per elaborare il programma utente ed il computer penserà solo... a se stesso!

Come fare, quindi, per dirottare la routine di Interrupt e fare in modo che elabori nostre routines l.m. lunghe e complesse?

Semplice: è sufficiente "spezzare" tali routines ed elaborarle una alla volta, interruzione dopo interruzione. Supponendo di voler trattare due routines (*alfa* e *beta*) troppo lunghe per essere elaborate in un solo ciclo, alla prima richiesta di Interrupt verrà elaborata *alfa*, alla successiva *beta*; poi di nuovo *alfa* e così via.

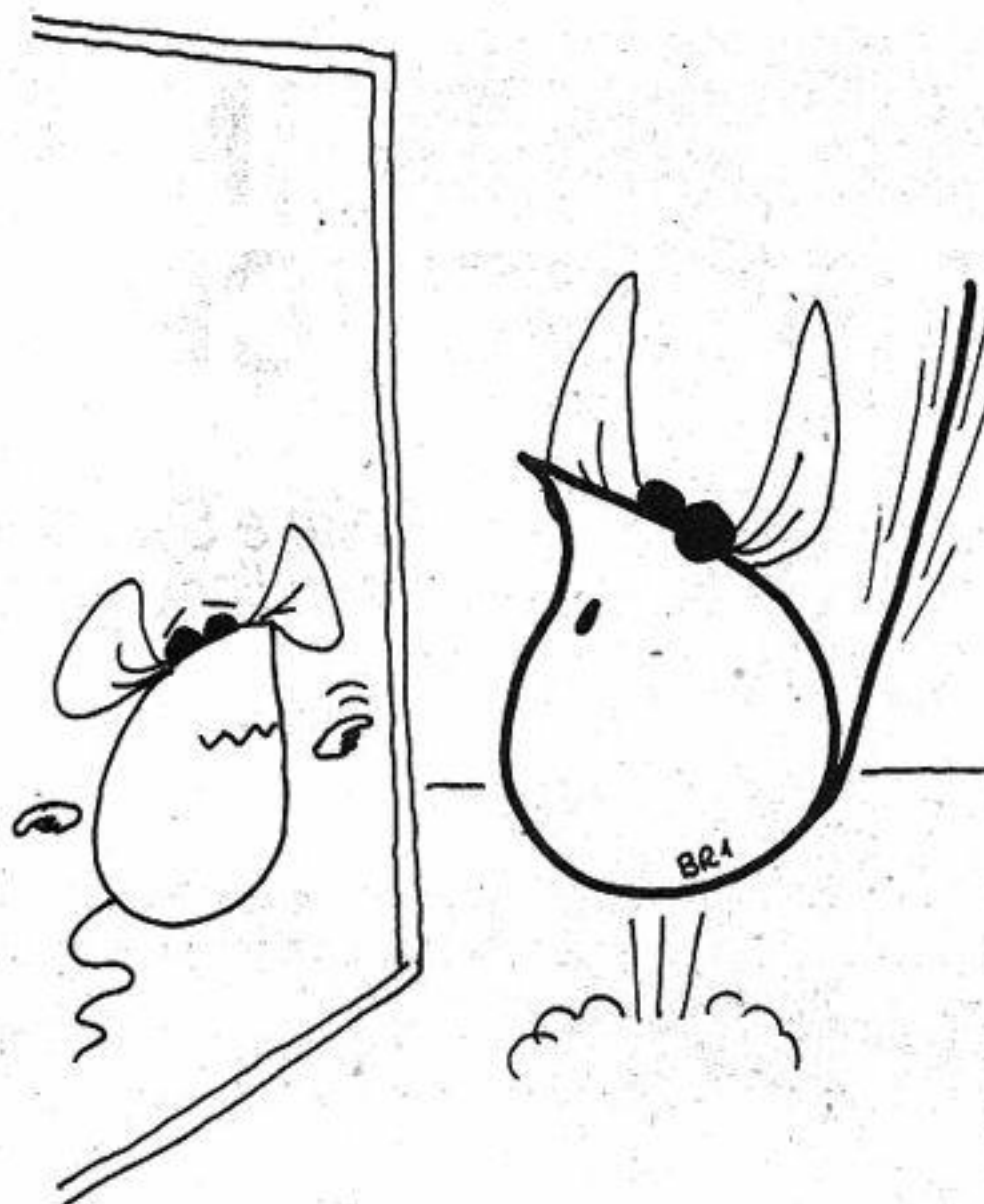
Una qualsiasi locazione di memoria Ram può funzionare da "deviatore".

```
1 rem ************************************
2 rem **          border clock          **
3 rem **               di               **
4 rem **paolo monti & edmondo fichera**
5 rem **          copyright 1990        **
6 rem ************************************

10 print"■";:printtab(8)"← aspetta 399"
15 print"caricamento dati"
20 a=49152+2048:b=53248+(48*8):q=0
25 fori=0to(64*11):pokea+i,0:next
30 poke56334,0:poke1,peek(1)and251
35 fore=0to10:forw=0to7
37 pokea+(64*e)+(w*3),peek(b+w+(e*8))
38 nextw,e
40 poke1,peek(1)or4:poke56334,129
45 c=0:fori=0to399:reada:print"■"i
47 poke49152+i,a:c=c+a:next
50 ifc<>49507thenprint"errore":end
55 si=54272:pokesi+24,15
57 pokesi+1,240:pokesi+6,240
60 poke49152+2046,42:poke49152+2047,42
65 print"■"
70 input"inserire orario (hhmmss)";h$
75 iflen(h$)<>6then70
80 er=0:gosub180:ifer<>0then70
85 gosub200
90 print"■"
95 input"inserire sveglia (hhmmss)";h$
100 iflen(h$)<>6then95
110 er=0:gosub180:ifer<>0then95
120 poke56335,peek(56335)or128:gosub200
```

```
122 poke56335,peek(56335)and127
125 sys49152
130 print"■premi 's' per far ";
133 print"partire l'orologio"
135 geta$:ifa$=""then135
140 ifa$="s"then150
145 goto135
150 poke56328,0:print"ok!":print
179 end
180 fori=1to6:z$(i)=mid$(h$,i,1)
182 z(i)=val(z$(i)):next
185 hr=z(1)*10+z(2)
187 ifhr<0orhr>23thener=1:return
190 m=z(3)*10+z(4):s=z(5)*10+z(6)
192 ifm<0orm>59ors<0ors>59thener=1:return
195 return
200 poke56329,z(5)*16+z(6)
205 poke56330,z(3)*16+z(4)
210 pm=0:ifhr>=12thenpm=128:hr=hr-12
213 ifhr=0thenhr=12
215 q1=int(hr/10):q2=hr-(q1*10)
220 poke56331,pm+(q1*16)+q2
225 return
997 rem    ****************************
998 rem    ***   data routine l.m.   ***
999 rem    ***                       ***
1000 data 120,169,157,141,020,003,169
1001 data 192,141,021,003,169,001,141
1002 data 026,208,173,017,208,041,127
1003 data 141,017,208,169,248,141,018
1004 data 208,169,000,170,168,185,129
1005 data 193,153,000,208,138,200,153
```

```
1006 data 000,208,200,192,016,208,240
1007 data 169,005,157,039,208,232,224
1008 data 008,208,248,169,255,141,023
1009 data 208,141,029,208,141,021,208
1010 data 169,000,133,002,133,163,133
1011 data 164,162,000,160,016,169,003
1012 data 133,001,189,000,208,072,169
1013 data 004,133,001,104,157,000,208
1014 data 232,208,237,238,088,192,238
1015 data 097,192,136,208,228,169,055
1016 data 133,001,169,208,141,088,192
1017 data 141,097,192,169,148,141,000
1018 data 221,169,196,141,136,002,169
1019 data 000,141,255,255,169,128,141
1020 data 014,220,169,147,032,210,255
1021 data 169,042,141,254,199,141,255
1022 data 199,088,096,173,017,208,041
1023 data 247,141,017,208,162,048,202
1024 data 208,253,009,008,141,017,208
1025 data 165,002,240,003,032,072,193
1026 data 173,013,220,041,004,240,004
1027 data 169,001,133,002,173,011,220
1028 data 170,016,010,169,001,133,252
1029 data 169,002,133,253,016,006,169
1030 data 000,133,252,133,253,138,041
1031 data 127,201,018,208,002,169,000
1032 data 032,054,193,032,068,193,024
1033 data 101,252,141,253,199,138,032
1034 data 068,193,024,101,253,201,042
1035 data 144,011,056,233,010,141,252
1036 data 199,238,253,199,016,003,141
1037 data 252,199,173,010,220,032,054
1038 data 193,032,068,193,141,251,199
1039 data 138,032,068,193,141,250,199
1040 data 173,009,220,032,054,193,032
1041 data 068,193,141,249,199,138,032
1042 data 068,193,141,248,199,173,008
1043 data 220,169,001,141,025,208,076
1044 data 049,234,133,251,041,015,170
1045 data 165,251,041,240,074,074,074
1046 data 074,096,024,105,032,096,230
1047 data 163,165,163,201,003,176,007
1048 data 169,017,141,004,212,016,041
1049 data 201,006,176,007,169,016,141
1050 data 004,212,016,006,201,009,176
1051 data 245,144,232,165,163,201,050
1052 data 144,018,230,164,165,164,201
1053 data 011,144,006,169,000,133,002
1054 data 133,164,169,000,133,163,096
1055 data 136,000,120,000,088,000,072
1056 data 000,040,000,024,000,104,000
1057 data 056
1058 end
```

```
.;**************************************
.;**     DISASSEMBLATO BORDER-CLOCK   **
.;**               di                 **
.;** E. Fichera Douglas & Paolo Monti **
.;**          Copyright 1990          **
.;**************************************

.c000 sei
.     lda #$9d       ;modifica
.     sta $0314      ;i puntatori
.     lda #$c0       ;di IRQ
.     sta $0315      ;
.     lda #$01       ;abilita le
.     sta $d01a      ;interrupt di raster.
.     lda $d011      ;posiziona
.     and #$7f       ;l'interruzione
.     sta $d011      ;alla
.     lda #$f8       ;linea 248 ($f8)
.     sta $d012      ;dello schermo
.     lda #$00       ;azzera registri
.     tax            ;
.     tay            ;
.c021 lda $c181,y    ;legge dalla tabella.
.     sta $d000,y    ;e scrive la posiz. .
.                    ;'x' dello sprite   .
.     txa            ;azzera acc.        .
.     iny            ;                   .
.     sta $d000,y    ;scrive '0' nelle   .
.                    ;'y' dello sprite   .
.     iny            ;prossimo sprite    .
.     cpy #$10       ;ha finito?         .
.     bne $c021      ;no, salta          .
.     lda #$05       ;colora di verde    .
.c033 sta $d027,x    ;tutti gli sprites  .
.     inx            ;                   .
.     cpx #$08       ;ha finito?         .
.     bne $c033      ;no, salta          .
.     lda #$ff       ;                   .
.     sta $d017      ;espansione vert.   .
.     sta $d01d      ;espansione orizz.  .
.     sta $d015      ;abilita 8 sprites  .
.     lda #$00       ;                   .
.     sta $02        ;azzera avvisatore  .
.                    ;di allarme         .
.     sta $a3        ;azzera contatore   .
.                    ;50esimi di secondo .
.     sta $a4        ;azzera contatore   .
.                    ;dei 10 secondi di  .
.                    ;durata allarme     .
.     ldx #$00       ;prepara copia      .
.     ldy #$10       ;caratteri          .
.c052 lda #$03       ;abilita ROM        .
.     sta $01        ;caratteri          .
.c056 lda $d000,x    ;legge dato caratt. .
.c059 pha            ;e lo salva         .
.     lda #$04       ;riabilita zona RAM .
.     sta $01        ;da $d000 a $ffff   .
.     pla            ;riprende dato      .
.c05f sta $d000,x    ;lo scrive in RAM   .
.c062 inx            ;dato successivo    .
```

```
        bne $c052   ;controlla se ha
                    ;finito la 'pagina'
        inc $c058   ;incrementa 'pagina'
                    ;di lettura
        inc $c061   ;incrementa 'pagina'
                    ;di scrittura

        dey         ;ha finito?
        bne $c052   ;no, salta
        lda #$37    ;ripristina il
        sta $01     ;valore di default
        lda #$d0    ;ripristina 'pagina'
        sta $c058   ;in lettura
        sta $c061   ;e in scrittura
        lda #$94    ;abilita banco 3
        sta $dd00   ;del VIC
        lda #$c4    ;sposta l'editor
        sta $0288   ;di schermo nel
                    ;banco 3
        lda #$00    ;azzera ultima
        sta $ffff   ;locazione banco 3
        lda #$80    ;spegne il
        sta $dc0e   ;TIMER A
        lda #$93    ;pulisce lo schermo
        jsr $ffd2   ;(SHIFT+CLR/HOME)
        lda #$2a    ;posiziona gli
        sta $c7fe   ;sprites 7 e 8 sui
        sta $c7ff   ;'due-punti'
        cli         ;
        rts         ;
```

```
         ROUTINE SOTTO INTERRUPT
                    ;
 Routine che fa scomparire il bordo
                    ;
.c09d lda $d011     ;setta il modo
      and #$f7      ;24 righe
      sta $d011     ;
      ldx #$30      ;ciclo di ritardo
.c0a7 dex           ;
      bne $c0a7     ;
      ora #$08      ;risetta il modo
      sta $d011     ;25 righe
                    --------
      lda $02       ;controlla se deve
                    ;far suonare
                    ;l'allarme
      beq $c0b6     ;no, salta avanti
      jsr $c148     ;si', esegue routine.
.c0b6 lda $dc0d     ;legge registro di
                    ;controllo interrupt.
      and #$04      ;e' scattato
                    ;l'allarme?
      beq $c0c1     ;no, prosegue
      lda #$01      ;si',attiva registro.
      sta $02       ;avvisatore allarme
.c0c1 lda $dc0b     ;legge registo ore
      tax           ;lo pone in x
      bpl $c0d1     ;salta se e' AM
      lda #$01      ;scrive una decina
      sta $fc       ;nel registro decine.
```

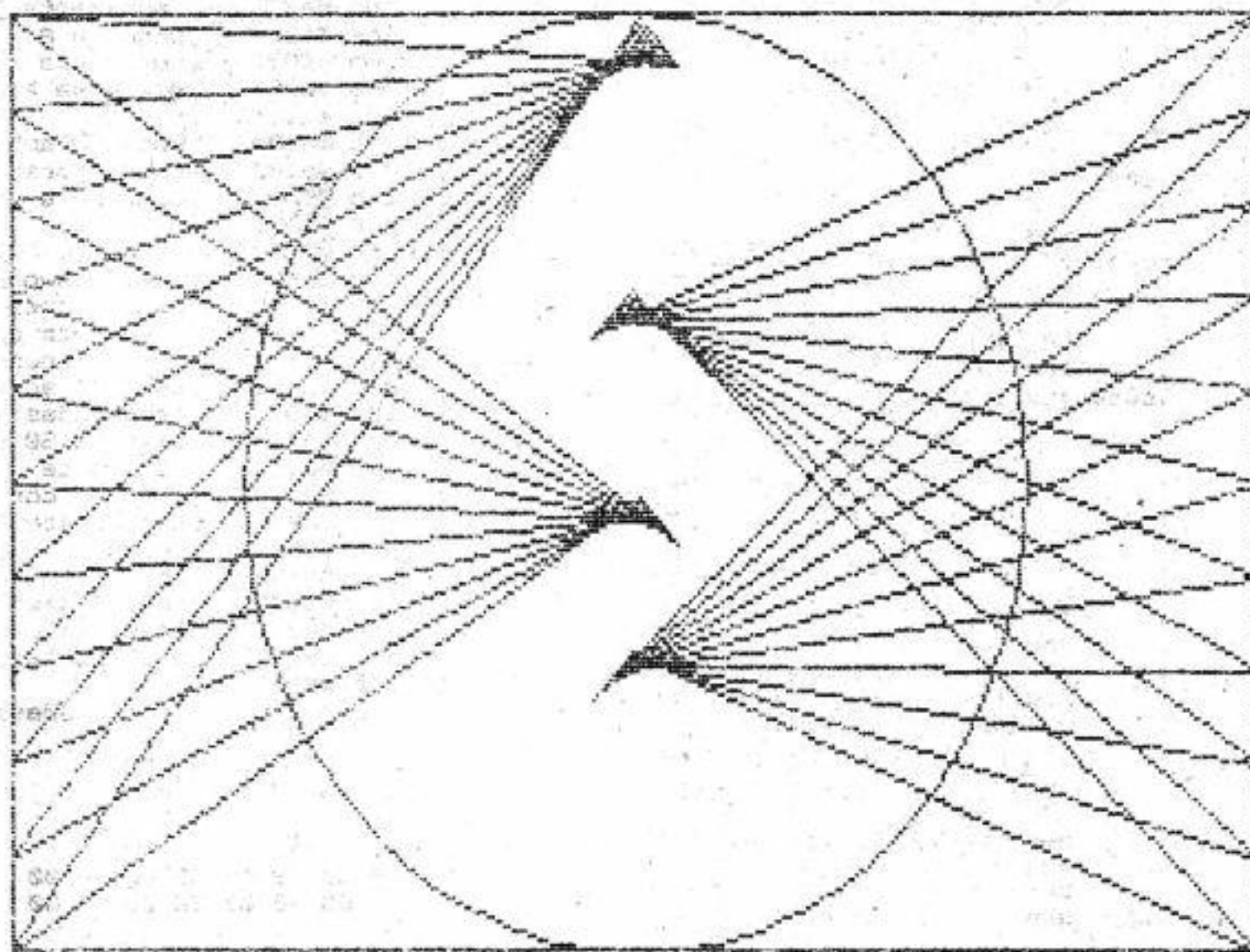
In effetti, con la vittoria di Iceman a G.P.C., il volo si conclude in modo poco ortodosso... Nei giorni successivi, tuttavia, si svolgono voli di addestramento dall'esito certamente più brillante, così da far ben sperare il buon Primo sull'esito della gara. E così arriva il grande giorno...

```
.        lda #$02     ;scrive due unita'  .
.        sta $fd      ;nel registro unita'.
.        bpl $c0d7    ;salto forzato      .
.c0d1    lda #$00     ;scrive zero        .
.        sta $fc      ;in decine          .
.        sta $fd      ;e unita'           .
.c0d7    txa          ;riprende ora       .
.        and #$7f     ;azzera AM/PM       .
.        cmp #$12     ;confronta con le 12.
.        bne $c0e0    ;salta se e' diverso.
.        lda #$00     ;                   .
.c0e0    jsr $c136    ;salta alla routine .
.                     ;che separa decine e.
.                     ;unita' in A e X    .
.        jsr $c144    ;salta alla routina .
.                     ;che aggiunge $20   .
.        clc          ;                   .
.        adc $fc      ;aggiunge o 0 o 1   .
.        sta $c7fd    ;scrive in decine   .
.        txa          ;prende unita'      .
.        jsr $c144    ;aggiunge $20       .
.        clc          ;                   .
.        adc $fd      ;aggiunge o 0 o 2   .
.        cmp #$2a     ;confronta con le 10.
.        bcc $c102    ;salta se > di 10   .
.        sec          ;altrimenti         .
.        sbc #$0a     ;sottrae 10         .
.        sta $c7fc    ;e scrive in unita' .
.        inc $c7fd    ;e incrementa decine.
.        bpl $c105    ;salto forzato      .
.c102    sta $c7fc    ;scrive unita'      .
.c105    lda $dc0a    ;legge i minuti     .
.        jsr $c136    ;separa             .
.        jsr $c144    ;aggiunge $20       .
.        sta $c7fb    ;scrive decine      .
.        txa          ;prende unita'      .
.        jsr $c144    ;aggiunge $20       .
.        sta $c7fa    ;scrive unita'      .
.        lda $dc09    ;legge secondi      .
.        jsr $c136    ;                   .
.        jsr $c144    ;                   .
.        sta $c7f9    ;scrive decine      .
.        txa          ;                   .
.        jsr $c144    ;                   .
.        sta $c7f8    ;scrive unita'      .
.        lda $dc08    ;sblocca il latch   .
.        lda #$01     ;conferma           .
.        sta $d019    ;l'interrupt        .
.        jmp $ea31    ;                   .
.                     ;                   .
.c136    sta $fb      ;salva registro     .
.        and #$0f     ;separa le unita' e .
.        tax          ;le pone in X       .
.        lda $fb      ;riprende registro  .
.        and #$f0     ;separa decine      .
.        lsr          ;le sposta sul      .
.        lsr          ;nibble             .
.        lsr          ;di destra          .
.        lsr          ;                   .
.        rts          ;ritorna            .
.                     ;                   .
.c144    clc          ;aggiunge $20 per   .
.        adc #$20     ;puntare all'area   .
.        rts          ;sprites            .
.                     ;                   .
.        SUBROUTINE DI SUONERIA           .
.                     ;                   .
.c148    inc $a3      ;incrementa 50esimi .
.        lda $a3      ;pone in A i 50esimi.
.        cmp #$03     ;confronta con 3    .
.        bcs $c157    ;salta se > di 3    .
.c150    lda #$11     ;                   .
.        sta $d404    ;accende suono      .
.        bpl $c180    ;salto forzato      .
.c157    cmp #$06     ;confronta con 6    .
.        bcs $c162    ;                   .
.c15b    lda #$10     ;                   .
.        sta $d404    ;spegne suono       .
.        bpl $c168    ;salto forzato      .
.c162    cmp #$09     ;confronta con 9    .
.        bcs $c15b    ;se e' > non suona  .
.        bcc $c150    ;se e' < suona      .
.c168    lda $a3      ;legge 50esimi      .
.        cmp #$32     ;siamo a 50 50esimi?.
.        bcc $c180    ;no, salta          .
.        inc $a4      ;increm. contatore  .
.        lda $a4      ;ha suonato per 11  .
.        cmp #$0b     ;secondi?           .
.        bcc $c17c    ;no, salta          .
.        lda #$00     ;se si' azzera      .
.        sta $02      ;l'avvisatore       .
.        sta $a4      ;e il contatore     .
.c17c    lda #$00     ;                   .
.        sta $a3      ;azzera 50esimi     .
.c180    rts          ;                   .
.
.
.TABELLA
.
.c181 88 00 78 00 58 00 48 00
.c189 28 00 18 00 68 00 38 00
```

HARD COPY C/128, IN BASIC E' PIU' VELOCE

Una brevissima routine in Basic 7.0, senza Peek, Poke e Sys, che batte in praticità (e velocità) le complesse procedure in linguaggio macchina

di Fabrizio Asruffi

Uno dei motivi che fanno pensare al C/128 come ad un personal computer di livello superiore a quello del C/64 è il fatto che possiede un interprete più evoluto, il Basic 7.0 che, tra l'altro, permette di effettuare disegni in alta risoluzione molto facilmente.

Nondimeno alcuni lo definisco "limitato", ed è vero. Una di queste limitazioni, ad esempio, è il fatto che non esiste un'istruzione specifica per trasferire la pagina grafica su carta, cosa molto utile soprattutto se si tratta di diagrammi, scritte, grafici, etc.

Una routine in Basic può essere veloce (quasi) come una complessa routine il linguaggio macchina

La routine di queste pagine, priva di Sys, Poke e Peek, si adatta ad essere "trasportata" in vostri programmi Basic

Di conseguenza si è costretti ad arrangiarsi per conto propio, riccorrendo a metodi più o meno complessi, ma che in basic si sono sempre rivelati, fino a ieri, lentissimi.

Cosicchè, per avere risultati accettabili, bisognava ricorrere a brevi ma complicate routine in L.M..

Ma siamo sicuri che siano proprio le più veloci ?

LA TEORIA

Abbiamo spesso parlato delle caratteristiche con cui la stampante permette di ricevere e stampare i codici grafici (ampliamente e in modo dettagliato sul C.C.C. numero 50, di conseguenza diamò per scontate quelle informazioni, riprendendo solo i punti principali nel caso sia neccessario.

GSHAPE E SSHAPE

Passiamo alla descrizione delle istruzioni *Sshape / Gshape*, la cui sintassi è ormai conosciuta da tutti, ma nessuno si è mai chiesto come facciano a memorizzare una zona del grafico in una variabile stringa.

Per quanto riguarda l'istruzione *Sshape* (*Gshape* è la corrispondente funzione opposta, ma si basa sullo stesso principio), una volta impartita, dobbiamo immaginare che suddivida l'area del grafico indicata in tante righe (ognuna alta un pixel) le quali, a loro volta, sono suddivise in tanti settori da otto pixel partendo da sinistra.

A questo punto, tenendo presente che a pixel acceso corrisponde *1* e a pixel spento corrisponde *0*, muovendosi proprio come se stesse leggendo, memorizza i vari settori con un codice compreso da 0 a 255 e li memorizza nella variabile stringa sotto forma di codice Ascii. Nel caso in cui le righe siano costituite da un numero di pixel non multiplo di 8, quando l'istruzione legge l'ultimo settore di ogni riga (che sarà, quindi, costituito da un numero di pixel minore (7, 6, 5, .., 1)), lo legge comunque come un codice "normale", dal momento che suppone alla sua destra un numero 0 in grado di far sembrare anche quel settore costituito da 8 pixel.

Quando verrà chiesto di ridisegnare, in quello stesso punto o in un altro, quell'area (tramite l'istruzione Gshape), non terrà conto degli zeri aggiunti, altrimenti potrebbe danneggiare un altro disegno eventualmente già presente. Per far ciò, ma non solo per questo, l'istruzione Sshape, dopo aver terminato di memorizzare la zona indicata, in fondo alla stringa aggiunge le coordinate relative del riquadro memorizzato: ..., *X1*, *X*, *Y1*, *Y*, in cui *X* e *Y* sono le coordinate relative dell'angolo superiore sinistro (solitamente, quindi, corrispondono a 0, 0), mentre *X1* e *Y1* sono quelle dell'angolo opposto.

IL PROGRAMMA

A questo punto basta fare 1+1, ed è proprio quello che il programma presentato effettua.

Nella prima parte esegue un semplice disegno, il quale può essere sostituito con un qualsiasi altro grafico da voi effettuato, purchè generato in Graphic 1.

In seguito procede memorizzandolo nel vettore *A$( )*. Tramite il ciclo di riga 230 viene memorizzato l'intero disegno, suddividendolo in fasce di 8 x 200 pixel.

Come si può vedere, però, il ciclo ha uno step di soli 7 pixel per volta, dovuto al fatto che la testina della stampante, come è noto, può stampare solo fasce di un altezza massima di 7 dots. Tuttavia deve ricevere codici di 8 bit con il più significativo (MSB), sempre posto a 1. Ecco, quindi, che si giustifica la presenza della riga 240, la quale, prima di memorizzare ogni area, traccia una linea sui primi pixel di questa. Inoltre il fatto che lo step sia di soli 7 pixel per volta, fa sì che la linea tracciata non "rovini" il disegno. Infatti viene segnata sull'ultimo pixel della fascia precedente, ormai già memorizzata nel vettore.

Una volta memorizzato il disegno, viene richiesto a quale periferica lo si vuole inviare,

se alla stampante o al video (i simboli in reverse presenti prima e dopo la *V* e la *S* presenti nel *Print* di riga 280, servono per cambiare il colore del cursore da giallo a bianco e viceversa). Per quanto riguarda l'output su video, in pratica si mette in atto lo stesso ciclo utilizzato per la memorizzazione, evidentemente eseguito al contrario, senza però dover tracciare alcuna linea.

Per quanto riguarda l'output sulla stampante, prima di inviare le varie celle del vettore occorre escludere le coordinate relative presenti in fondo a ciascuna di essa.

A questo punto è bene elencare alcune note relative al listato.

Dato che le fasce memorizzate dal video sono in posizione verticale (mentre la stampante le dispone sul foglio orizzontalmente), il disegno che si otterrà sulla carta risulterà ruotato di 90 gradi rispetto a quello che appare sul video.

Purtroppo la prima striscia di pixel dell'intero disegno, avente coordinate X=0, andrà persa, a meno che...

1- prima del ciclo di riga 230 si memorizzino i pixel in questione, ad esempio in B$:
Sshape B$, 0, 0, 0, 199

2- dopo lo stesso ciclo, magari a riga 255, si pulisca lo schermo (con *Graphic 1, 1*) e di seguito si eseguano le tre istruzioni:
Gshape B$, 7, 0
Draw 1, 0, 0 TO 0, 199
Sshape C$, 0, 0, 7, 199

3- per completare la visualizzazione sul video si riscriva la riga 350 fino a *Next*, poi alla 351...
Gshape B$, 0, 0
...e alla 352 quello presente sulla 350, ma solo da *Color 1, 2* in poi.

4- anche per quanto riguarda la stampa, sulla linea 400, tra *Next* e *Slow* si deve aggiungere l'istruzione
Print #4, Left$ (c$, 200).

Naturalmente si può anche pensare di adattare il listato per lo schermo multicolor (*Graphic 3*), tenendo però presente che per ogni pixel acceso valgono due bit, più precisamente:

00	color 0	sfondo
01	color 1	primo piano
10	color 2	multicolor 1
11	color 3	multicolor 2

Un ultimo consiglio è di far girare il programma, le prime volte, eliminando le istruzioni *Fast* per capire meglio il principio su cui si basa il listato.

PROPOSTE

La routine pubblicata, interamente in Basic, è forse valida anche per il **C/16** e **Plus/4** dal momento che i due interpreti sono molto simili al Basic 7.0.

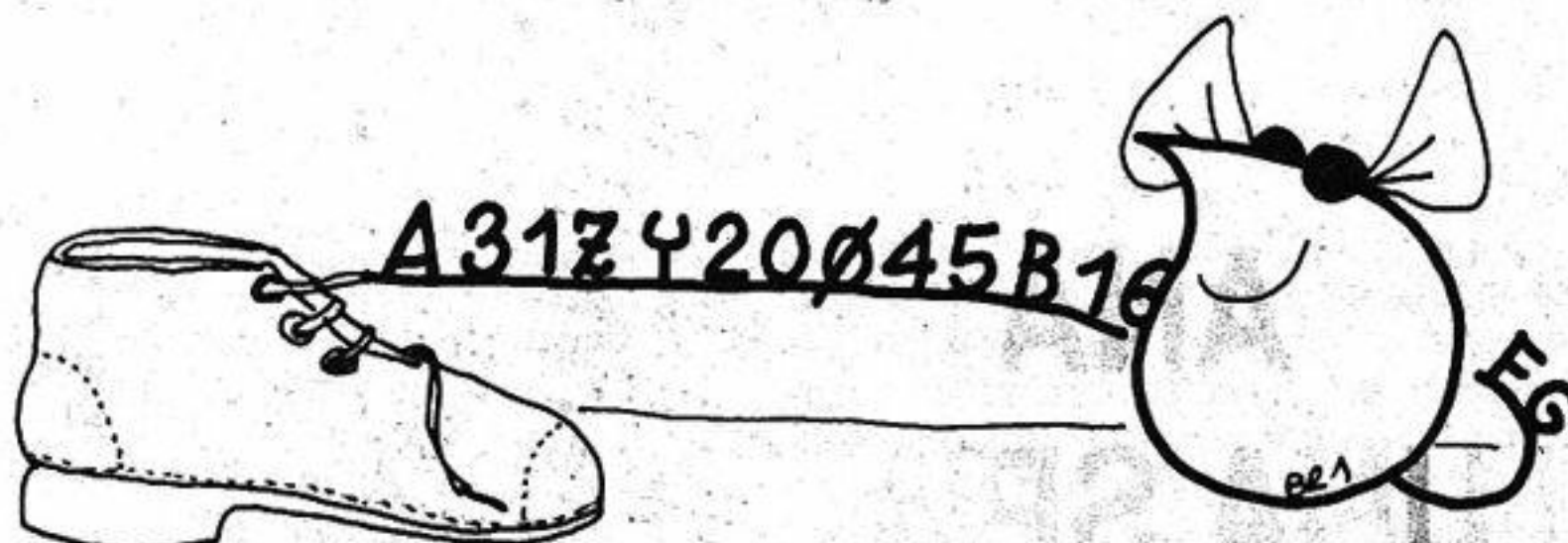
La velocità di esecuzione consente di ottenere una copia su carta in appena un minuto (contro un minuto e venti secondi della routine L.M. presente su C.C.C. 52).

Una routine in LM, inoltre, è piuttosto difficile da modificare o adattare ad altri programmi, specie se questi già contengono segmenti in l.m. Il programma risulta ottimo per stampare un'intera pagina grafica. Se, però, si modificano opportunamente i valori di inizio e di fine dei cicli di memorizzazione, visualizzazione e stampa (e se si modifica la lunghezza e la posizione di partenza delle fasce di schermo inserite nelle rispettive celle del vettore) si può prendere in considerazione qualsiasi porzione di schermo.

Scrivendo un programma specifico dovrebbe esser possibile disegnare, e gestire, aree di proporzioni maggiori, suddividendole in più zone (es. 640 x 400 costituita da quattro zone da 320 x 200 pixel ciascuna, oppure costituita da nove zone ciascuna di 256 x 160 pixel in grado di riempire l'intero foglio di carta). Si potranno ottenere effetti "speciali" grazie alle modalità offerte dall'istruzione Gshape (*normale, reverse, or, and, xor*).

Provate, inoltre, ad invertire o modificare le coordinate relative con cui salvate un'area di schermo; oppure ad invertire la sola sequenza dei codici memorizzati.

Alcune semplici modifiche, suggerite nell'articolo, potranno rendere la routine stessa ancora più versatile

Il listato, che contiene anche un semplice demo, si può adattare facilmente al C/16 ed al Plus/4 grazie alla notevole similittudine esistente tra i due interpreti

```
10 rem *-------------------------------*
20 rem *   hard copy graphic 1 c/128   *
30 rem *===========> by <==============*
40 rem *        fabrizio  arsuffi      *
50 rem *-------------------------------*
60 :
70 trap 410
80 :
90 rem esempio grafico <==========================================
95 rem inserire - fast - per una maggiore velocita' operativa
100 color4,7:     :color0,7:graphic1,1:color1,8:color0,7:color5,8
110 x=0:y=160:x0=58:y0=160:x1=102:x2=135
120 for u=200 to 0 step -20:i=int(u)
130 x=x+1:y=y+1:y0=y0-1:x0=x0+1:x1=x1+1:x2=x2+1
140 draw 1,0,i to y,x
150 draw 1,320,i to y0,x0
160 draw 1,0,i to y,x1
170 draw 1,320,i to y0,x2:next
175 draw 1,1,0 to 319,0 to 319,199 to 1,199 to 1,0
177 circle 1,160,100,100
180 :
190 rem inizio della routine hard copy graphic 1 <==================
200 :
210 rem memorizzazione della pagina grafica <=======================
220 dim a$(45)
230 for i=0 to 315 step 7:ii=i/7
240 draw 1,i,0 to i,199
250 sshape a$(ii),i,0,i+7,199:next
260 :
270 rem richiesta della periferica su cui inviare l'output <==========
280 graphic0,1:print"▓▓su ▓v◆ideo o su ▓s◆tampante ?":slow
290 getkey a$:if a$="v" then 330
300 if a$="s" then 380
310 goto 290
320 :
330 rem visualizzazione del graphic 1 sul video <====================
335 rem inserire - fast - per una maggiore velocita' operativa
340 :      :graphic1,1:color1,8:for i=45 to 0 step -1:ii=i*7
350 gshape a$(i),ii,0:next:color1,2:char 1,14,24,"premi un tasto":slow
360 getkey a$:fast:goto 270
370 :
380 rem stampa del graphic 1 su carta <============================
385 rem inserire - fast - per una maggiore velocita' operativa
390 fast:open 4,4:cmd4:print#4,chr$(8);
400 for i=45 to 0 step -1 :print#4,left$(a$(i),200):next:slow:goto 270
410 slow:end
```

C.C.C. BEST PLAYERS

LEVEL	NAME	PTS.
9	P.G	084550
5	J.A	059600
1	KAN	000800
1	RYU	000600
1	STE	000400
1	OHE	000200

ANATOMIA DI UNA SPROTEZIONE

Descriviamo una procedura, elaborata da un "Campione del Sowtware" già noto ai nostri lrttori, che illustra le fasi occorrenti per sproteggere il gioco originale. Power Drift servendosi di quella straordinaria macchina che è il C/128

di Luca Viola

Era un pomeriggio cupo e piovigginoso quando finalmente il mio negoziante di fiducia annunciò che era arrivato il tanto atteso *Power Drift Originale* su disco.

Dopo 15 minuti ero già a casa e mi apprestavo a far partire il programma dopo aver messo, come di consuetudine, la tacchetta di protezione sul dischetto. Fatto partire il programma, e dopo averci giocato un pò per testarlo, mi accingevo a fare la tradizionale copia di sicurezza, quando mi accorsi che il mio *Turbo Nibbler* preferito, inseparabile compagno di tante avventure... pirata, non era in grado di creare una copia funzionante del disco.

Questo mi suonava quasi come un insulto, e decisi che a tutti i costi avrei avuto la mia copia di sicurezza di Power Drift, avessi dovuto esaminare il dischetto originale bit per bit. Dato che comunque si era fatto tardi, decisi di dormirci su: la notte porta consiglio.

LA SFIDA

L'indomani mi alzai deciso a tutto, e incominciai ad esaminare il disco con vari *Disk-Scanner* e *Disk-Monitor*, senza venir a capo di nulla.

Sconsolato, stavo per rassegnarmi, e sovrappensiero fissavo il foglietto delle istruzioni presenti nella confezione originale, quando all'improvviso una nota presente sul foglietto attirò la mia attenzione: vi era scritto, infatti, testualmente:

Questo gioco, una volta caricato, non prevede ulteriori caricamenti sia da cassetta che da disco.

Ecco la soluzione che cercavo! Nella foga di vedere come era stato protetto il disco avevo tralasciato un'altra possibile strada: quella di agire sulla memoria del computer, registrandola, cioè, per intero dopo che il gioco fosse stato caricato, e dopo aver chiaramente individuato l'indirizzo di partenza del programma.

Inoltre, per quante protezioni vi possano essere, almeno il primo programma della Directory deve poter essere caricato normalmente con un comando del tipo *Load "*", 8,1*.

Se fossi riuscito a seguire passo-passo le azioni svolte dal computer durante il caricamento, sarei riuscito senz'altro a rintracciare gli indirizzi iniziale e finale del gioco, in modo da registrarlo senza problemi su un mio disco.

Il C/128, il mio computer, ben si presta alle azioni di pirateria, avendo due banchi separati di memoria *Ram*, il secondo dei quali completamente libero da interferenze. Per cui resettai in modo 128 e il primo passo compiuto fu quello di dare il comando *Directory* (se la si carica in modo 64 con *Load "$", 8* e successivo *List* non sarà visibile).

Mi fu svelata la presenza di due files: "*As*" e "*Gm1*".

Il file *As* era quello che veniva caricato con Load "*", 8, 1, e perciò era anche il responsabile del caricamento del resto del gioco, dato che eseguiva un *Autostart*. Allora presi il fido programma *Load Address* presente sul *1541-Test Demo Disk*, per verificare a quale indirizzo di memoria il file *As* venisse caricato. Il computer rispose *$02A7*. A questo punto, per poterlo meglio esaminare, lo caricai in *Bank 1*, non però a partire dal suo indirizzo reale (perchè ciò avrebbe sconvolto le delicate routine di sistema presenti in quell'area), ma dall'indirizzo *$12A7* anzichè *$02A7*. Questo per riuscire meglio a comprendere il flusso del programma: se avessi incontrato, esaminandolo, delle istruzioni del tipo *Jmp* oppure *Jsr*, avrei sostituito lo *zero* iniziale con un *uno* e avrei potuto interpretare correttamente il significato dei salti. Impartito, così, da monitor il comando...

L "As", 8, 112A7

... e caricato in memoria tale file, impartii il fatidico *D 112A7*.

Una serie di dati confusi apparì sul video. Provando con *M 112A7* fu chiaramente riconoscibile un miniprogramma Basic contente l'istruzione *Sys 2061*, che entrava in azione quando si fosse caricato il file As senza il suffisso "*,8,1*" ma semplicemente col "*,8*", caricandolo, cioè, in area Basic: un modo elegante di ottenere l' autostart usando entrambe le forme sintattiche di Load.

```
5 :rem ** go64 in ram 1 v2.0 **
6 :rem ** by viola luca 1990 **
7 :rem ** computer c/128 **
8 :
10 scnclr: bank 15
20 print "reading data..."
21 print "sys 4864 per andare in c/64"
30 for t = 4864 to 4915
40 read x$: x = dec(x$)
50 poke t, x: ck = ck + x
60 next
70 if ck <> 6559 then print "error"
71 :
100 data 20, e1, a7, f0, 05, c9, 59, f0, 01, 60
110 data a2, 00, bd, 1a, 13, 9d, 63, 02, e8, e0
120 data 1b, d0, f5, 4c, 63, 02, a9, 7e, 8d, 00
130 data ff, a9, 44, 8d, 06, d5, a9, e3, 85, 01
140 data a9, 2f, 85, 00, a9, f7, 8d, 05, d5, 4c
150 data e2, fc
160 end
```

Pensai, inoltre, che quando si digita il "*,1*" alla fine di un comando Load, il file stesso viene caricato a partire dalle stesse locazioni in cui era stato salvato.

Per realizzare l'autostart, quindi, era necessario che il programma *As* si estendesse almeno fino a *$302 - $303 (770 - 771)* in modo da modificare il vettore di Warm Start e farlo puntare all' indirizzo di partenza.

Diedi così il comando *M 11302*, che rivelò che nel vettore di Warm Start veniva inserito il valore *C4 - 02* (in forma low - high, quindi l'indirizzo reale era *$02C4*, a cui veniva ceduto il controllo al termine del caricamento di As).

LA PROTEZIONE VACILLA

A questo punto il comando D 112C4 aprì la prima porta: dal disassemblato, in cui venivano usate le routine kernal di *Setfiles, Setnam* e *Load* si evinceva che veniva caricato in maniera standard un altro file; in seguito il controllo gli veniva ceduto tramite un *Jmp $0334*.

La routine Setnam, infatti, puntava ad un nome di lunghezza 3 posto all'indirizzo $2C1; pertanto con M 112C1 riuscii a vedere il nome di tale file: come mi aspettavo, era ***Gm1***.

A questo punto controllai, tramite il programma *Load Address*, l'indirizzo di caricamento di Gm1, che, guarda caso, era proprio $334. Lo caricai in memoria (come al solito in bank 1) a partire dall'indirizzo $1334 anzichè $0334 - per meglio esaminarlo- con...

L "Gm1", 8, 1133

...dopo aver però spento e riacceso il computer per cancellare la memoria ed evitare confusioni con i dati del file caricato precedentemente.

Impartendo D 11334 la prima

UNA DOVEROSA PRECISAZIONE

Tutte le informazioni riportate nel presente articolo sono da ritenersi puramente didattiche. Qualsiasi uso improprio (soprattutto se a scopo di lucro) è da ritenersi Illegale. L'autore del presente articolo, e la Systems Editoriale, declinano pertanto ogni responsabilità per un uso non corretto delle note contenute nel presente articolo.

istruzione presente era Jmp $0353. Continuai quindi con D 11353. Apparì una tavola di Jsr ($36E-$37D), seguite da una serie di istruzioni di scrittura nei registri del Cia2 ($393-$3Aa).

COMPARE IL TURBO

Disassemblando i vari indirizzi delle Jsr mi accorsi che venivano modificati i *vettori* di Load: questo programma, quindi, era il turbo che provvedeva a velocizzare il caricamento di Power Drift.

Continuando a disassemblare la routine principale notai agli indirizzi $3B0-$3D5 che veniva richiamata per 6 volte, con in accumulatore un numero crescente da 0 a 5, una subroutine sita a $3F9.

Impartendo D 113F9 vidi che tale subroutine conteneva un salto alla routine di load, a cui andava dopo aver settato alcuni puntatori: essa doveva essere quindi incaricata di caricare i vari pezzi come, per esempio, la schermata grafica che appare all'inizio.

Seguivano agli indirizzi $3D7 - $3Eb i salti ad alcune routine kernel per chiudre la comunicazione col drive, e a $3Ed una Jsr a $6D5.

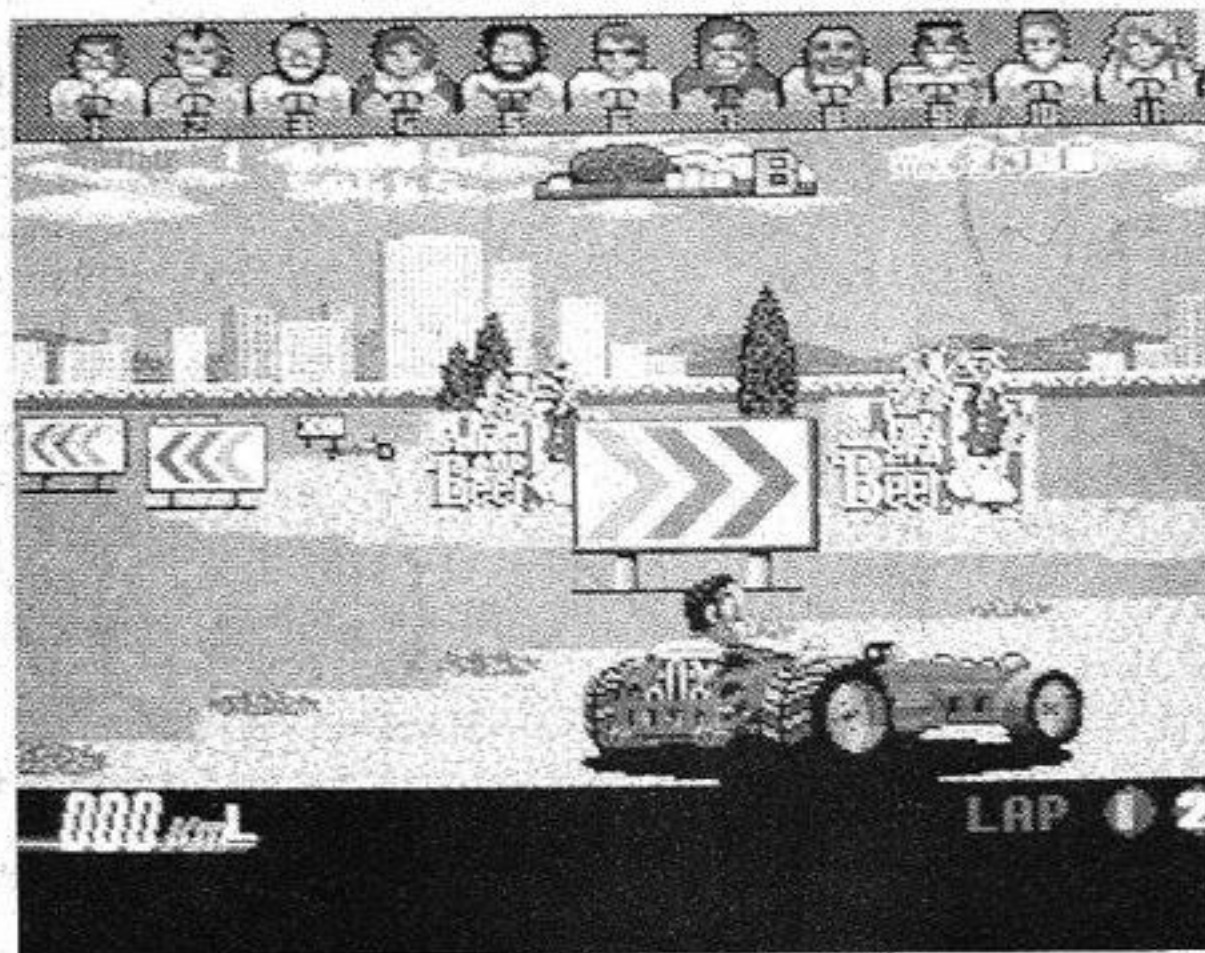
Impartendo D 116D5 mi accorsi interpretando il disassemblato che tale subroutine effettuava una *decodifica* dei dati caricati in turbo.

La routine principale, infine, chiudeva con un salto a $0834 (+2100 in decimale): **avevo trovato l'indirizzo di partenza del programma!**

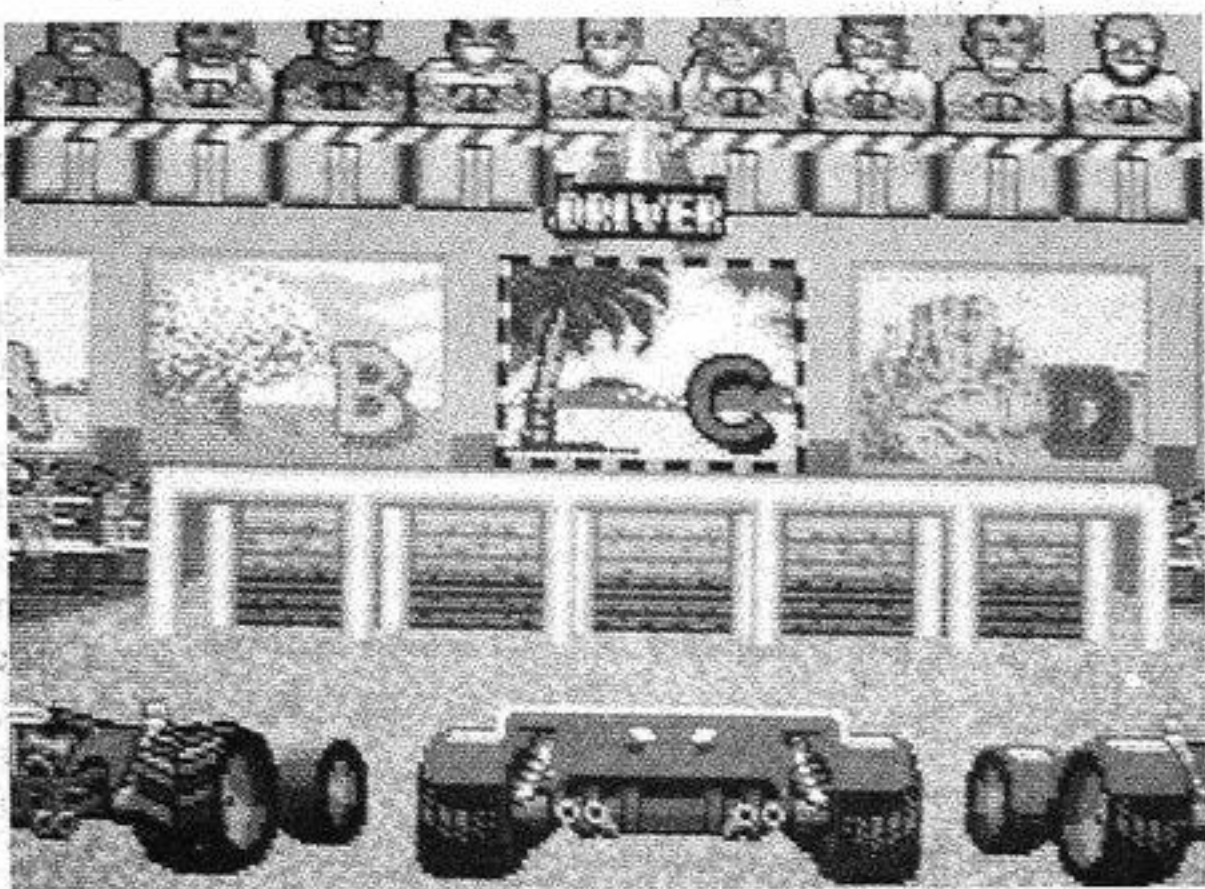
Mi restava da scoprire dove finisse il programma.

Per far ciò riempii la memoria del bank 1 col codice *BD* (un altro codice comunque sarebbe andato bene) usando il comando...

F 10400 1FFFF BD

Questo perchè, una volta caricato il gioco, avrei potuto resettare e vedere col comando *M* dove in memoria ricominciasse la sequenza di *Bd*: quello sarebbe stato l'indirizzo finale del codice di Power Drift.

Uscito quindi dal monitor, caricai il mio programmino Go64 in modo da andare in modalità C/64 attivando la Ram 1 anzichè 0: così facendo al momento del reset mi sarei ritrovato nel bank 1 il codice di Power Drift perfettamente integro.

Caricai l'originale con Load "*", 8, 1

Una volta partito, mi accorsi che effettivamente il turbo si fermava per 6 volte durante il caricamento, e questo avvalorò la mia ipotesi che il codice presente da $3B0 a $3D5 servisse a caricare 6 files tramite la subroutine allocata a $3F9.

Terminato il caricamento fu effettuata la decodifica e non appena il gioco partì, resettai.

Tornato in modo 128, diedi un...

M 1F000

...per trovare dove finisse il codice di Power Drift e ricominciasse la sequenza di codici *BD* che avevo posto in memoria prima di andare in modo 64.

Non trovando nulla, impartii M 1F400, poi M 1F800, finchè, dando, M 1Fa00 mi accorsi che a partire da $1Fa10 ripartiva la sequenza di codici Bd: il gioco dunque terminava a $Fa10.

A questo punto avevo tutti i dati necessari: salvai quasi tutta la memoria del bank 1 con...

S "Power Drift", 8, 10400, 1Fa10

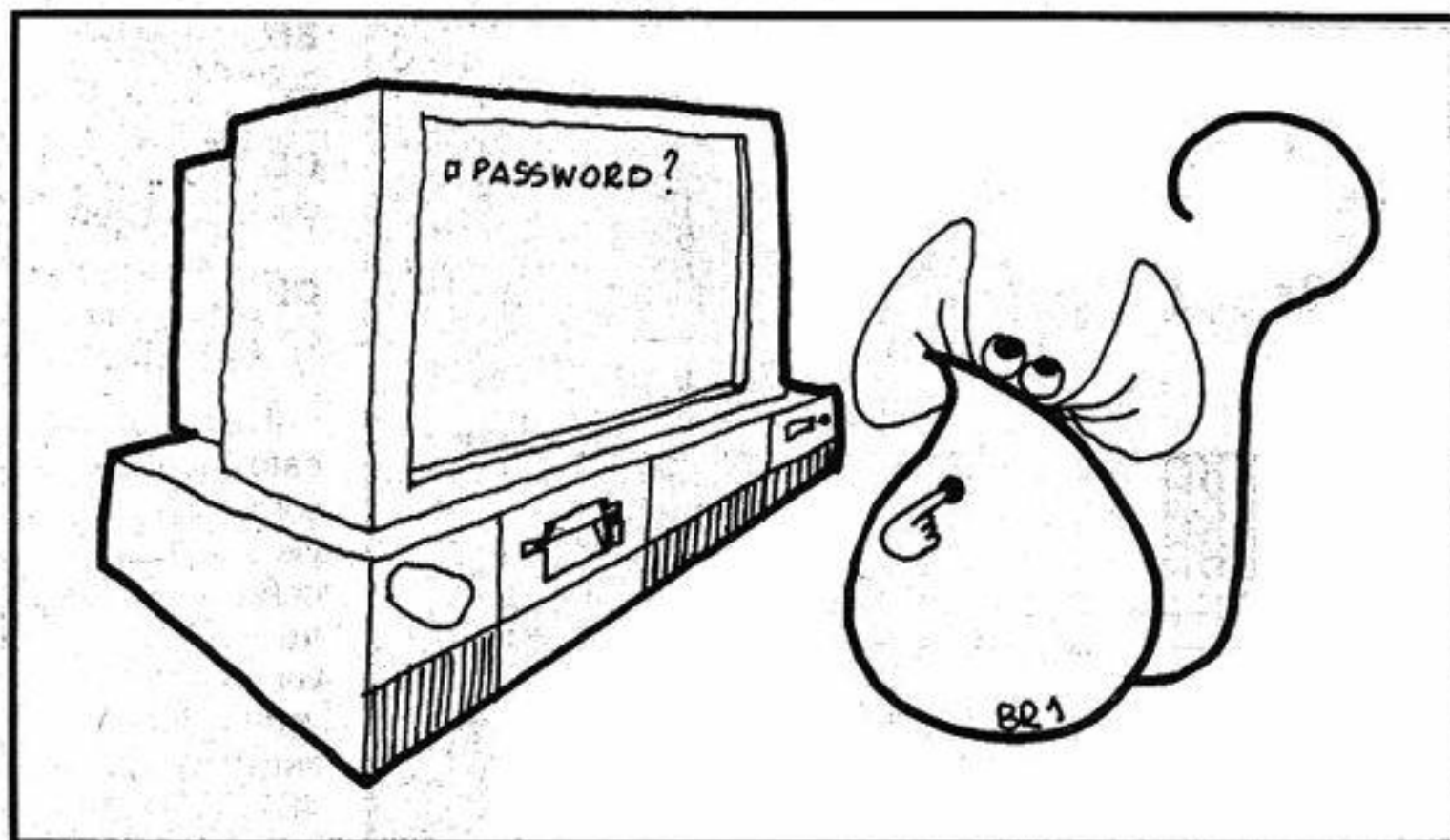
Il file registrato, comunque, era troppo lungo e non direttamente caricabile in modo 64 (ben 250 blocchi!). Si rendeva necessario compattarlo: in giro esistono diversi compattatori in grado di agire su file così grandi.

Uno dei miei, *File Compressor III*, può compattare fino a 250 blocchi (che fortuna!) e chiede l'indirizzo di memoria a cui saltare dopo aver decompattato il file: a questa domanda risposi senz'altro $0834 (che era l'indirizzo di partenza di Power Drift ricavato poco prima esaminando il codice del file Gm1, ricordate?).

Sottoposi il (lungo) file alla procedura di compattazione, e dopo una discreta attesa, otteni un file compattato della lunghezza di 200 blocchi (il limite massimo di un file caricabile in modo 64 è di 202 blocchi).

A questo punto tornai in modo 64, caricai il file così ottenuto, e, con comprensibile trepidazione, diedi il run... ***Funzionava!*** (quasi quasi mi dispiaque per Chris Butler, ma tant'è...).

E così, dopo 2 ore e mezza circa di lavoro, ero riuscito ad ottenere la mia copia di sicurezza, ma non ero del tutto tranquillo.

In serata, infatti, il negoziante a cui mi ero rivolto mi aveva garantito l'arrivo di *Turbo Out Run...*

TUTTE LE FINESTRE CHE VOLETE

Una semplice procedura in Gw-Basic offre una nuova possibilità per abbellire i nostri programmi

di **Luca Morettoni**

Molti di noi si saranno divertiti almeno una volta con il basic di Amiga ad aprire finestre grafiche; oppure hanno visto quanto siano più intuitivi da usare programmi che "dialogano" con l'utente attraverso le già citate finestre.

Una volta, però, tornati al vecchio C/64, ci ritroviamo di fronte al solito (e tetro) schermo a quaranta colonne.

Cerchiamo ora di spiegare (anche se sommariamente) come vengono gestite le finestre da Amiga; ed i problemi che abbiamo, invece, con il C/64 per aprire una finestra.

LE FINESTRE

Le finestre vengono gestite da Amiga come se fossero veri e propri schermi indipendenti, senza però perdere ciò che ciascuna finestra sembra "coprire" momentaneamente. Il miracolo avviene trasferendo, ciò che viene occultato, in una zona di memoria protetta da altre scritture. In seguito, i dati vengono richiamati solo quando la finestra viene chiusa nuovamente.

Tale lavoro viene effettuato in frazioni di secondo attraverso sofisticati circuiti integrati di nuova concezione, tra cui il famoso Blitter.

PROBLEMI...

Naturalmente il C/64 (a causa soprattutto alla sua età) non brilla certo come velocità operativa e quindi bisogna un po' arrangiarsi.

Per realizzare finestre grafiche con il C/64 è stata subito accantonata l'idea di trasferire in altre zone Ram l'area grafica coperta dalle finestre.

L'accorgimento citato è stato scelto non tanto per la velocità (decisamente accettabile, se realizzato in linguaggio macchina) ma per la limitata memoria disponibile: provate voi a sistemare, in soli 50 KRam, il Gw-Basic, lo schermo grafico, un programma ed una manciata di finestre (ricordo che una finestra a tutto schermo occupa ben 8K!).

Il problema viene quindi risolto trasferendo "frammenti" di schermo su disco per poi essere ricaricati quando si chiude una finestra. La velocità operativa subisce ovviamente un calo, ma la quantità di finestre apribili aumenta (quasi...) all'infinito.

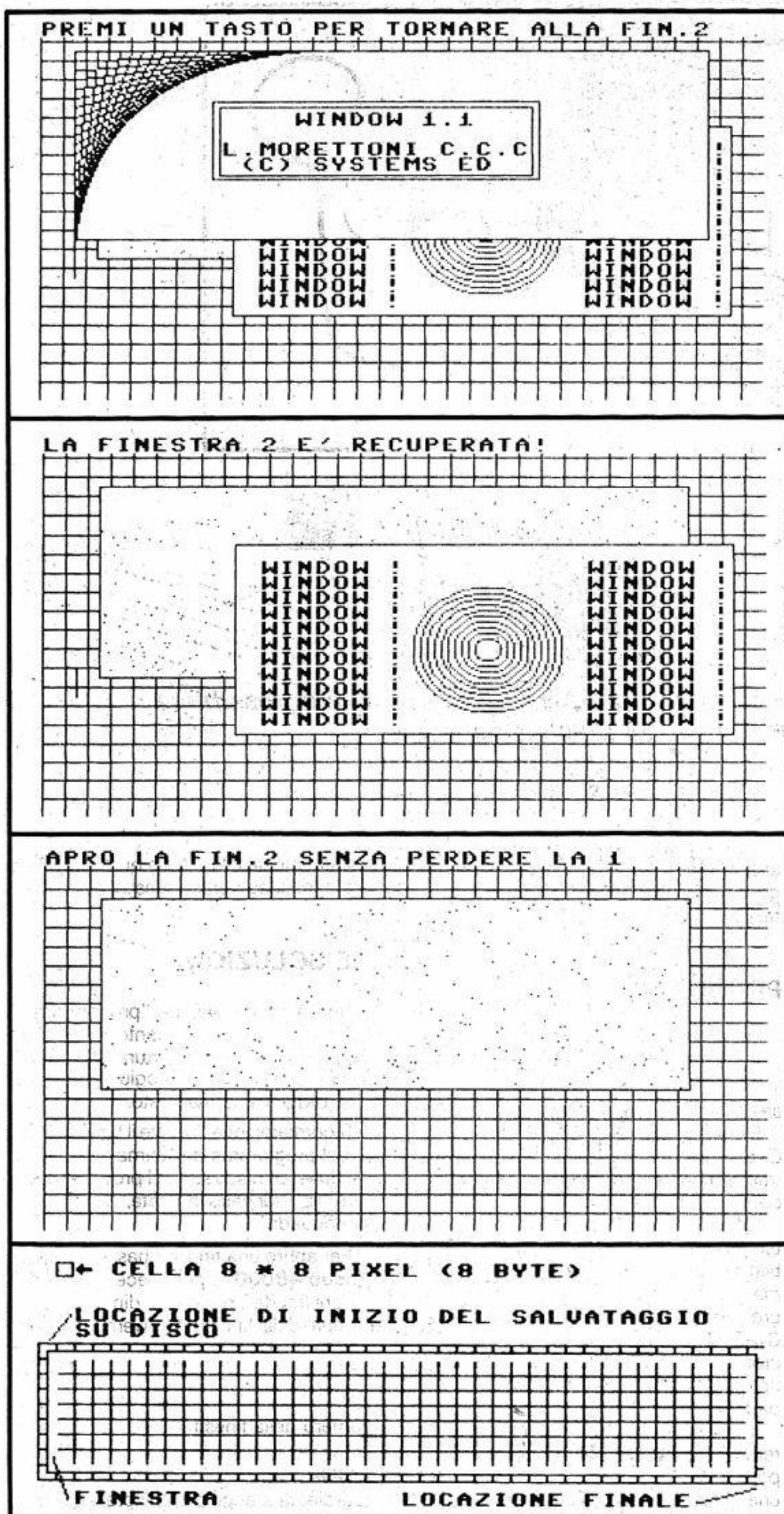
...E SOLUZIONI

Non è esatto definire "programma" la soluzione adottata, in quanto sono soltanto due le subroutine (apertura e chiusura finestre) che si possono aggiungere a qualsiasi programma realizzato.

Ricordiamo che durante il funzionamento del programma deve rimanere inserito, nel drive, un disco su cui il programma salverà (e, successivamente, caricherà) i vari riquadri.

Per aprire una finestra basta l'istruzione **Gosub 48000**; è però necessario, prima di "chiamarla", passare le dimensioni ed il numero della finestra attraverso le seguenti variabili:

NF%
numero della finestra

AAX%
coordinata x angolo superiore sinistro

BBY%
coordinata y angolo superiore sinistro

CCX%
coordinata x angolo inferiore destro

DDY%
coordinata y angolo inferiore destro

Il numero della finestra deve essere indicato sempre in ordine cresente, pena l'insorgere di vari problemi in fase di chiusura (se una finestra era già aperta, il vecchio contenuto viene sostituito dal nuovo); inoltre, se le coordinate non sono corrette, vengono controllate dal programma ed eventualmente modificate (in alcuni casi le finestre non vengono aperte, vedi linee 48110 - 48200).

Per la chiusura delle finestre, invece, la procedura è decisamente più semplice. Basta indicare il numero della finestra (variabile NF%) e saltare alla routine con **Gosub 48410**: il programma provvederà a caricare la porzione di schermo al posto che le compete.

ALCUNE PRECISAZIONI

E' bene sottolineare, a scanso di equivoci, che prima che il C/64 apra una finestra sullo schermo occorre un tempo proporzionale all'area della finestra stessa. Segue ora una breve nota sul funzionamento delle due subroutine.

La finestra, pur appartenendo allo schermo grafico, viene individuata in una griglia di **320 x 25** (come se fosse in modo testo).

Se ne determina quindi l'indirizzo iniziale nella pagina grafica a partire da $E000 (decimale 57344) e viene calcolata la sua occupazione in numero di byte (variabili LCI e BYO). La porzione di schermo viene quindi salvata in un file binario (istruzione **Bsave**, riga 48300) per poi essere ricaricata con l'istruzione **Bload**.

Il file contenente lo sfondo viene cancellato dopo la chiusura della finestra (riga 48500).

Tutte le porzioni di schermo registrate hanno come nome il numero della finestra, preceduto da uno spazio, e come suffisso la sigla **.WND**.

Un ultimo suggerimento è quello di abbellire i bordi delle finestre, magari ponendovi in alto il nome corrispondente (vedi linee 48340 - 48350).

PER CHI INIZIA

Il listato di queste pagine è stato scritto usando il "dialetto" Gw-Basic che differisce dal Basic standard del C/64 in numerose istruzioni.

Chi non possiede nè il drive nè il dischetto **Gw-Basic Emulator** (che è possibile tuttavia richiedere al nostro servizio arretrati, vedi ultime pagine di questo fascicolo) non può quindi esaminare i simpatici "effetti" del programma

Chi, invece, è pratico di linguaggio macchina, sarà sicuramente in grado di esaminare la tecnica usata dal nostro collaboratore e "trasferirla" sul C/64, pur se questo è privo dell'emulatore.

Coloro che saranno in grado di realizzare finestre usando altri linguaggi (o altri computer) possono telefonare in Redazione (02/52.49.211) nei pomeriggi di lunedì e giovedì di ogni settimana per concordarne l'eventuale pubblicazione.

```
100 REM WINDOW DEMO !
110 :
120 KEYOFF:CLS1:SCREEN1
130 FORA=0TO320STEP10
140 LINE(0,A)-(320,A)
150 LINE(A,0)-(A,200):NEXT
160 LOCATE 1,1
170 PRINT"ORA APRO LA FIN.1 SENZA PERDERE IL FONDO"
180 AAX%=25 :BBY%=25
190 CCX%=285:DDY%=125:NF%=1
200 GOSUB48000
210 LOCATE 1,1
220 PRINT"PUNTI CASUALI NELLA FINESTRA N.1        "
230 FORS=1TO3:FORA=25TO285
240 PX=INT(RND(0)*100)+25
260 PSET(A,PX):NEXTA,S
270 LOCATE1,1
280 PRINT"APRO LA FIN.2 SENZA PERDERE LA 1"
290 AAX%=85 :BBY%=55
300 CCX%=305:DDY%=155:NF%=2
310 GOSUB48000
320 LOCATE1,1
330 PRINT"NELLE FIN. SI POSSONO TRACCIARE GRAFICI"
340 FORA=6TO33STEP3
350 CIRCLE(197,110),A:NEXT
360 LOCATE1,1
370 PRINT"OPPURE SCRIVERE TESTI ...               "
380 FORA=0TO10
390 LOCATEA+9,13:PRINT"WINDOW !"
400 LOCATEA+9,31:PRINT"WINDOW !":NEXT
410 AAX%=15 :BBY%=15
420 CCX%=295:DDY%=135:NF%=3
430 GOSUB48000
440 LOCATE1,1
450 PRINT"E PER FINIRE ...       "
460 FORA=15TO135STEP5
470 LINE(A,15)-(15,150-A):NEXT
480 LOCATE7,15:PRINT"WINDOW 1.1"
490 LOCATE9,11:PRINT"BY LUCA MORETTONI"
500 LOCATE10,12:PRINT"(C) SYSTEMS ED"
510 LINE(79,45)-(217,81),1,B
520 LINE(76,42)-(220,83),1,B
530 LOCATE1,1
540 PRINT"PREMI UN TASTO PER TORNARE ALLA FIN.2"
550 GETA$:IFA$=""THEN550
560 NF%=3:GOSUB48410
570 LOCATE1,1
580 PRINT"LA FINESTRA 2 E' RECUPERATA!           "
590 GETA$:IFA$=""THEN590
600 NF%=2:GOSUB48410
610 LOCATE1,1
620 PRINT"ORA ECCO LA FINESTRA 1 COME ERA PRIMA"
630 GETA$:IFA$=""THEN630
640 NF%=1:GOSUB48410
650 LOCATE1,1
660 PRINT"ANCHE LO SFONDO E' STATO RECUPERATO  "
670 GETA$:IFA$=""THEN670
680 SCREEN0:END
46000 :
47000 END
47010 REM *---------------------------*
47020 REM !                           !
47030 REM ! SUBROUTINE WINDOW  VER.1  !
47040 REM ! PER C/64 E GWBASIC VER.2  !
47050 REM !                           !
47060 REM !  IDEATO E REALIZZATO DA   !
47070 REM !       LUCA MORETTONI      !
47080 REM !                           !
47090 REM !   (C) 1989 SYSTEMS ED.    !
47100 REM !                           !
47110 REM *---------------------------*
47120 :
48000 REM SUB.WINDOW OPEN
48010 REM CHIAMARE CON 'GOSUB 48000'
48020 REM PARAMETRI RICHIESTI :
48030 REM NF%  : NUMERO FINESTRA
48040 REM AAX% : COORD.X.ANG.SUP.SIN
48050 REM BBY% : COORD.Y.ANG.SUP.SIN
48060 REM CCX% : COORD.X.ANG.INF.DES
48070 REM DDY% : COORD.Y.ANG.INF.DES
48080 :
48090 REM CONTROLLO COORDINATE
48100 :
48110 IF NF%=0 THEN RETURN
48120 IF AAX%<0 THEN AAX%=0
48130 IF AAX%>319 THEN AAX%=319
48140 IF CCX%<AAX% THEN RETURN
48150 IF CCX%>319 THEN CCX%=319
48160 :
48170 IF BBY%<0 THEN BBY%=0
48180 IF BBY%>199 THEN BBY%=199
48190 IF DDY%<BBY% THEN RETURN
48200 IF DDY%>199 THEN DDY%=199
48210 :
48220 REM CALCOLI DIM. FINESTRA
48230 :
48240 LCI=&HE000+(INT(BBY%/8)*320)
48250 BYO=((INT(DDY%/8)+1)*320)-(LCI-&HE000)
48260 :
48270 REM REFRESH SU DISCO
48280 :
48290 W$=STR$(NF%)+".WND"
48300 BSAVE W$,LCI,BYO
48310 :
48320 REM APRE LA FINESTRA
48330 :
48340 LINE (AAX%,BBY%)-(CCX%,DDY%),0,BF
48350 LINE (AAX%,BBY%)-(CCX%,DDY%),1,B
48360 :
48370 REM ESCE
48380 :
48390 RETURN
48400 :
48410 REM SUB.WINDOW CLOSE
48420 REM CHIAMARE CON 'GOSUB 48410'
48430 REM PARAMETRI RICHIESTI :
48440 REM NF%  : NUMERO FINESTRA
48450 :
48460 REM REFRESH DAL DISCO
48470 :
48480 W$=STR$(NF%)+".WND"
48490 BLOAD W$
48500 KILL W$
48510 :
48520 REM ESCE
48530 :
48540 RETURN
48550 END
```

COME FARSI CESTINARE ARTICOLI E PROGRAMMI

L'irrefrenabile desiderio di collaborare con la nostra rivista spesso fa dimenticare elementari norme di sopravvivenza

di Alessandro de Simone

Uno dei misteri più affascinanti, per chi lavora in una rivista, è certamente rappresentato dalla sfrenata fantasia di coloro che aspirano a diventare scrittori o "giornalisti".

La figura del giornalista, almeno secondo le credenze popolari più accreditate, è stata definitivamente smondanata fin dai tempi dei film in bianco e nero americani degli anni '50. In queste pellicole, un giovane scrivano del Connecticut (tradizionalmente deriso da familiari ed amici) riesce a realizzare uno scoop d'eccezione intervistando (a seconda della fantasia dello sceneggiatore holliwoodiano di turno) un marziano arrivato fresco fresco da Marte oppure, a scelta, fotografando il presidente degli Stati Uniti mentre tenta di vendere missili a Fidel Castro. Le conseguenze dello scoop diventano, in seguito, ordinaria routine: lo scrivano gode dell'invidia dei suoi ex-amici, si fidanza con la più carina del gruppo (che lo sfotteva più degli altri) e diventa direttore ipercontinentale del *New York Times*, con tanto di sigaro (cubano) in bocca.

Il lettore umano "medio" di un qualsiasi periodico (e con tale termine intendiamo colui che, oltre a leggere, sa anche scrivere) ritiene di non aver nulla da invidiare agli scrivani del Connecticut, anche se risiede, più modestamente, a Bernareggio e non gli capita di aver spesso a che fare con marziani o presidenti ma, al massimo, con il parroco o col farmacista.

Paradossalmente, però, il racconto nel cassetto, che ognuno di noi possiede (ci riferiamo al cassetto), non sempre viene fotocopiato ed inviato alla Garzanti o alla Mondadori ed il motivo è presto detto: chi ci assicura che, una volta inviato, la signora Garzanti (o la signorina Mondadori) non cambiano subdolamente il nome sulla copertina, appropriandosi di fama, onori e, soprattutto, soldi?

Per essere al riparo da brutte sorprese bisognerebbe andare da un notaio e depositare il manoscritto prima di inviarlo. E se poi il notaio è parente stretto della signora Garzanti (o è sposato con una Mondadori?)... Il mondo, si sa, è cattivo; è buona norma prudenziale aspettare che diventi più buono prima di inviare il manoscritto.

I PROGRAMMATORI

La sottospecie più feroce degli aspiranti giornalisti è rappresentata dai lettori di riviste tecniche, in generale, ed informatiche, in particolare.

Questi personaggi ritengono (nell'ordine) che: *1*- Tutto proviene dalla Tecnica e tutto finisce in essa. *2*- L'uomo che non vive di tecnica non è tale. *3*- Il mondo sarebbe migliore se privato di coloro che non vivono secondo le regole monastiche dei punti 1 e 2.

I programmatori, poi, sono le creature, in assoluto, più pericolose del pianeta, anche perchè non facilmente biodegradabili. Rimangono in uno stato di pericolo latente solo fino all'età di circa 14 anni, oppure fino a che non entrano in possesso di un calcolatore.

Il principale segnale del raggiungimento della maturità si evidenzia con il malcelato disprezzo dei cuccioli (che si arrabattono con il Basic alla meno peg-

gio) e nella esclusione violenta, dal branco, di coloro che non ricordano a memoria tutti gli indirizzi di tutte le Rom.

Ligio alle regole prima accennate, il programmatore - aspirante - giornalista ritiene, giustamente, che la Redazione al completo di una qualsiasi rivista di informatica trascorra il suo tempo ad attendere, trepidamente, che il postino consegni il suo straordinario programma (della serie *I Listati Che Hanno Cambiato Il Mondo*) e non si preoccupa minimamente dei trascurabili dettagli inerenti la comprensibilità, il funzionamento e l'uso del listato stesso.

I più furbi si limitano ad inviare un dischetto (naturalmente privo di etichetta) che, partendo in *autoboot*, visulizza meravigliosi demo e procedure forse interessanti, ma tutte rigorosamente in linguaggio macchina, magari crittografato, e senza listato sorgente, nè articolo esplicativo.

Come si possa, poi, stampare il listato è un mistero glorioso la cui soluzione viene affidata al basso volgo.

IN PRATICA

Allo scopo di non esser fraintesi, pertanto, è bene esser brutali per mettere in pre-allarme gli aspiranti collaboratori. Ecco, quindi, che cosa accade alle vostre lettere quando pervengono in Redazione.

1: Non appena giunge una qualsiasi missiva, la busta che la contiene viene immediatamente stracciata, bruciata e le sue ceneri sparse nel mare. Il motivo del selvaggio comportamento è dovuto unicamente al fatto che lo spazio a disposizione per archiviare la corrispondenza è piccolo (e la gente mormora) e tutto il "superfluo" deve essere eliminato.

2: Si esamina, subito dopo, il contenuto della busta; questa *doveva* contenere un dischetto ed un *unico* foglio di "accompagnamento".

Tutti i fogli in più (presi casualmente, tranne uno) vengono stracciati, bruciati e le loro ceneri gettate nel water (non possiamo andare al mare per ogni lettera: sarebbe antieconomico).

3: A questo punto (e solo a questo punto) si inizia a leggere la lettera. In questa, anzitutto, *deve* esserci l'intestazione: la *Systems Editoriale* pubblica ben 18 (diciotto) testate e non si può perder tempo a leggere la lettera per capire a quale rivista è indirizzata. Inoltre *deve* esser presente la data di invio, e questo nel vostro esclusivo interesse; verrà data precedenza, infatti, alle lettere che presentano data meno recente (non fate però i furbi datandole 1825, ce ne accorgeremmo subito).

Le lettere prive di data (e/o/u di intestazione) verranno poste in coda alle altre ed esaudite solo se c'è tempo (cioè mai). Nella lettera lasciate perdere i preamboli del tipo *"siete la rivista più forte del mondo"*: lo sappiamo già. Arrivate subito al sodo descrivendo sommariamente, ma chiaramente, il contenuto del dischetto. In fondo, poi, non dimenticate il vostro nome, cognome, indirizzo e, soprattutto, telefono!

4: Il dischetto allegato *deve* esser dotato di etichetta su cui *deve* comparire il nominativo completo dell'autore, la data di spedizione, il tipo di computer su cui "gira" ed il nome dei files presenti. La Directory del dischetto *deve* essere "pulita" (lasciate perdere i trucchetti relativi alla visualizzazione di ID multicolor animati, con sottofondo musicale: non ci impressionano più di tanto); *deve* essere inoltre priva di files inutili e superflui (tipo "separatori" e simili frattaglie). Il dischetto, soprattutto, **non** deve partire in autoboot; vi assicuriamo che siamo in grado di digitare *Load* e *Run*, nè ci costa fatica farlo.

Nel caso di articoli per Amiga, i dischetti *devono* contenere:

- l'articolo (in formato Ascii puro, semplice e spartano) *privo* del tutto di formattazioni particolari (caratteri sottolineati, neretto, cambi di font e simili barbarismi: ci pensiamo noi ad imbruttire il tutto).
- i vari programmi sorgenti (anch'essi in formato Ascii).

Onde evitare dispiaceri, provate a mandare in stampa articolo e programmi utilizzando (da *Shell*) il comando *Print:* se la stampante risponde, avete buone probabilità che il tutto sia valido.

Il dischetto, sempre nel caso di Amiga, non deve contenere le varie subdirectory **L**, **C** e simili. Il dischetto deve essere stato originariamente prodotto con il comando *Format* nella sua veste più discreta. Come fare, però (si chiderà l'astuto aspirante collaboratore) se il programma inviato richiede la presenza di tali subdirectory? E qui casca l'asino! Non dobbiamo dimenticare che la nostra rivista si rivolge soprattutto ai principianti ed a coloro che, di norma, non intendono ammalarsi di informatite. Dobbiamo immaginare che il nostro lettore "medio", dopo aver letto l'articolo, accende il computer e provvede a digitare il listato, registrandolo su di un dischetto che può essere *qualunque* e, come tale, anche privo dei files vitali eventualmente necessari per il buon funzionamento della procedura.

L'aspirante collaboratore, pertanto, deve tenere a mente tale eventualità facendo comparire nel listato, ad esempio, un messaggio del tipo *Print "Nel dischetto presente in DF0: deve esser presente la subdirectory L: in caso contrario premi Ctrl C e provvedi ad eseguire la copia di L:"*. Naturalmente tali avvertimenti devono esser presenti anche nell'articolo.

I dischetti che, ad una sommaria visualizzazione della directory, dovessero contenere altre directory, verranno infettati con la varicella e rispediti al mittente, senza francobollo.

PIANGE IL TELEFONO

Naturalmente la stessa procedura può (anzi, *deve*) essere seguita da coloro che intendano servirsi della nostra **BBS**. Preparate i vari files che costituiscono il vostro lavoro, dotateli di nomi non più lunghi di **otto** caratteri (per il nome) e **tre** (per l'eventuale suffisso).

Quest'ultimo deve essere:

.Asc (Articoli)
.Bas (Programmi Basic)
.Ass (Disassemblati)
.C (Sorgenti C)
.Let (Lettera di accompagnamento)

Non sapete ancora il numero della nostra BBS? Eccovi accontentati:

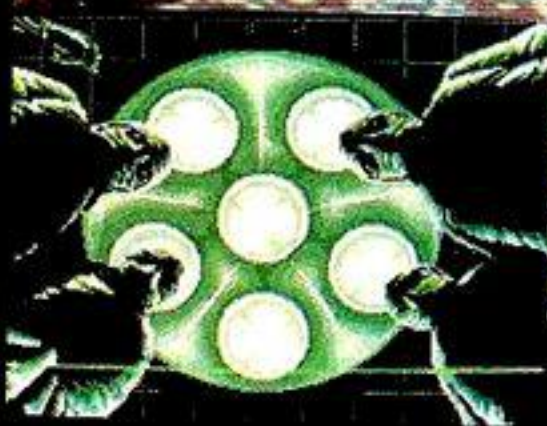
02 / 52. 49. 211

da 300 a 2400 baud; parità none; n. bits 8; un bit di stop. In altre parole: 2400, N, 8, 1.

La nostra BBS si adatta automaticamente alla velocità del vostro modem ed opera 24 ore su 24, sette giorni alla settimana, 12 mesi all'anno. Cioè sempre.

Commodore
COMPUTER
CLUB
La rivista degli utenti di sistemi Commodore

IRON LORD

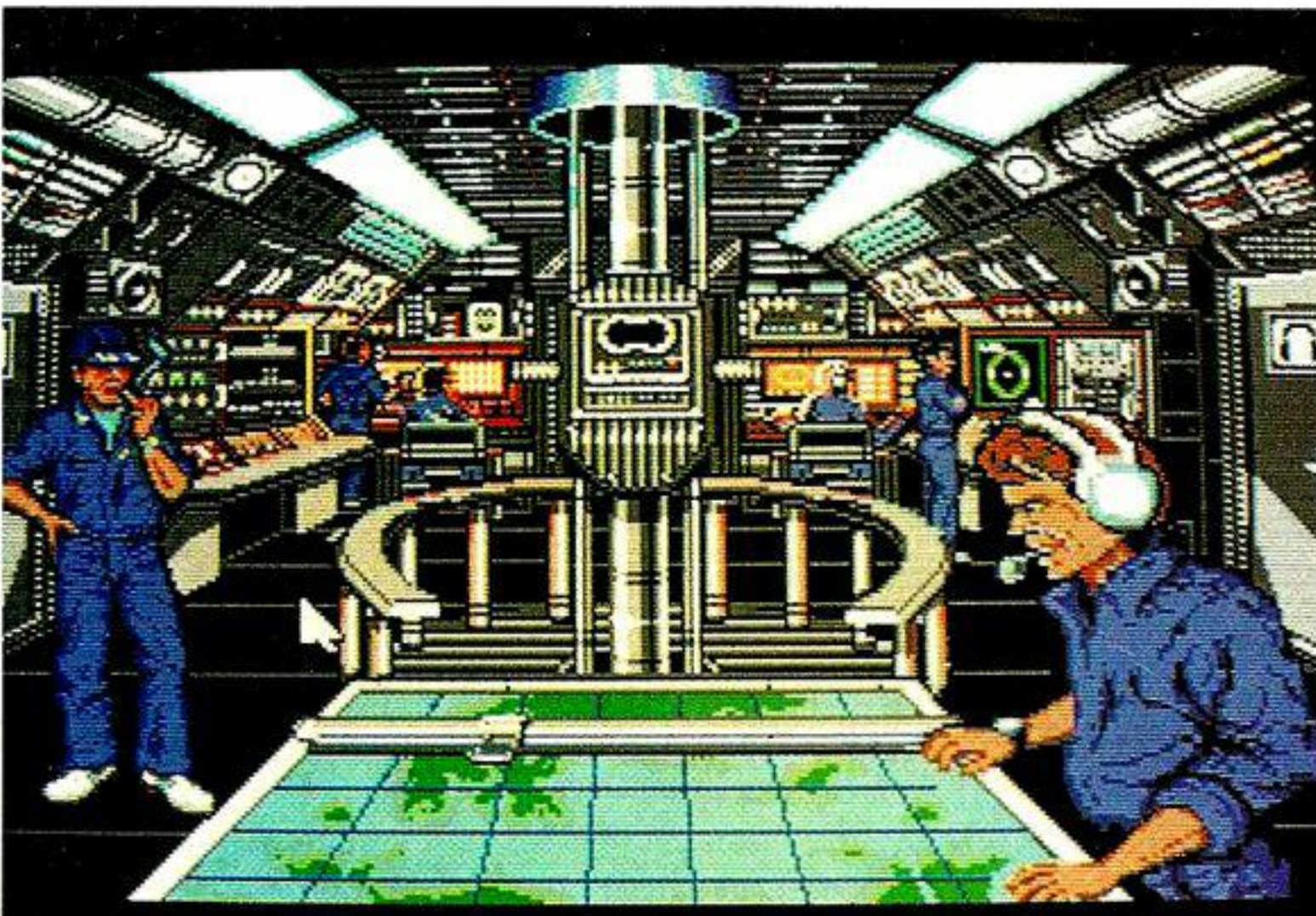
Mystery of the Mummy

AMIGA

Commodore MODEL 1081

POWER

688 SUB ATTACK

Su di un sottomarino a caccia di navi nemiche in uno scenario decisamente realistico

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Simulazione sottomarino
Softhouse: Electronic Arts

La simulazione di battaglie da bordo di un sottomarino conta un buon numero di titoli per Amiga *(Hunt for Red Hoctober, Sub Battle Simulator, Submarine Hunter, Submarine Simulator...)*, ma quando scende in campo la leggendaria Electronic Arts come minimo bisogna fare attenzione. Questo programma assomiglia tantissimo a *Sub Battle Simulator* perchè all'inizio si sceglie il quadro di controllo cui accedere indicando, con il mouse, un personaggio tra quelli raffigurati in un interno di sottomarino.

Per prima cosa bisogna richiedere, via radio, il tipo di missione (ve ne sono almeno sei, differenziati a loro volta in vari sottolivelli), poi si può iniziare a manovrare periscopi, radio, radar, sonar, motori, sistemi di puntamento, siluri, analizzare mappe e sagome di navi memorizzate nel computer di bordo.

Tutte le operazioni sono controllate via mouse clickando sui gadget visualizzati. A differenza di Sub Battle Simulator, qui troviamo un numero enorme di controlli e parecchie persone alle nostre dipendenze, che tra l'altro compaiono effettivamente come visi digitalizzati fedelmente sul video. Le missioni sono talvolta piuttosto complesse ed impegnative, ma come al solito è previsto un sistema di compressione del tempo per completare i lunghi inseguimenti in un tempo ragionevolmente breve. L'intero programma è comunque molto realistico ed appassionante, sebbene sia necessario studiare un ponderosomanuale d'uso per imparare ad usarlo a fondo.

LA TECNICA

Partendo dal peggio, gli effetti sonori avrebbero potuti essere migliori e di certo sono l'unico punto discutibile dell'(altrimenti) ottimo programma. I numerosissimi schermi comprendono le indicazioni di numerosi tipi di navi differenti, con sagome ed informazioni rigorosamente vere. I radar sono realistici, anche se la fase di costruzione delle immagini viste dal periscopio è un pochino lenta, così come le interfasi di passaggio da un pannello di controllo all'altro, che impongono la lettura di dati dal disco.

IL VOTO

Un altro piccolo capolavoro dalla ECA, con solo qualche difettuccio. 8-.

BUDOKAN

Ancora una simulazione di arti marziali, ma molto ben realizzato

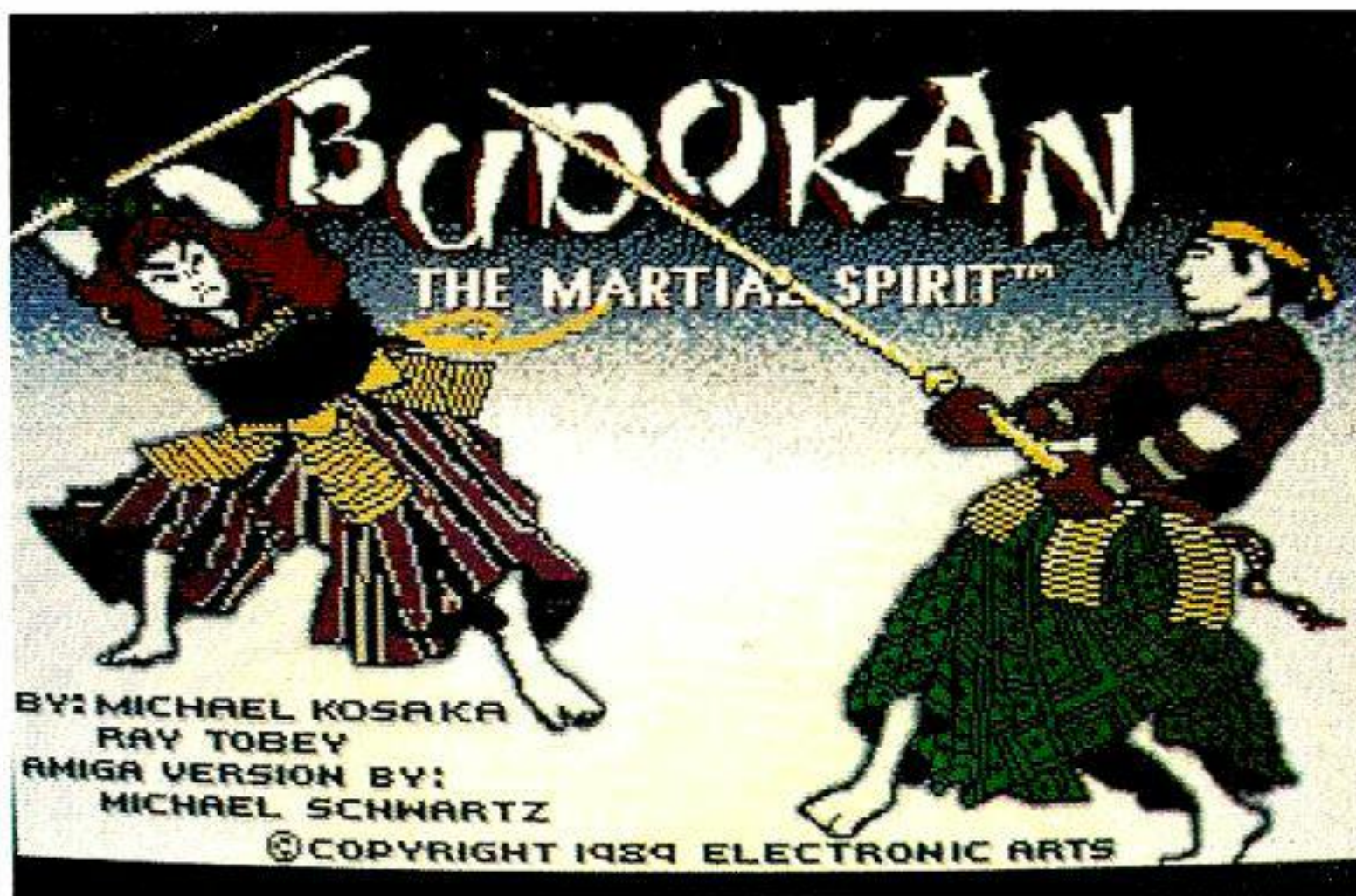
Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arti marziali
Softhouse: Electronic Arts

Un gioco di arti marziali (come *International Karate, Bruce Lee, The Way of Exploding Fist*...) non poteva che essere scritto da un orientale (*Michael Kosaka*) ed è certamente uno dei più evoluti nel suo genere, con ben quattro differenti specialità: *Karate, Kendo, Bo* e *Nunchaku.*

Per chi non fosse troppo... acculturato, precisiamo che il *Karate* si basa su colpi di mano e di piedi nudi, il *Kendo* su di una lotta con una spada (di legno),

il *Bo* su scontri a bastonate, e che il *Nunchaku* è uno strumento, originariamente usato dagli agricoltori giapponesi, trasformato in arma, consistente in due bastoncini legati da una corda (resi famosi da Bruce Lee).

Inoltre è prevista la possibilità di lottare in due avversari "umani" uno contro l'altro.

Inizialmente si sceglie tra le cinque pagode che contengono gli esperti delle quattro specialità dette prima (una è per scegliere la lotta a due), poi si indica, via joystick, un livello tra tre di abilità dell'avversario ed infine si può iniziare a lottare *(Kumite)* oppure scegliere la pratica *(Jiyu Renshu).*

Il controllo avviene come sempre tramite joystick: secondo lo spostamento della manopola e la pressione del tasto di fuoco si possono effettuare una notevole quantità di movimenti e colpi. Lo scopo è di diventare maestri in tutte le specialità, seguendo anche le osservazioni ed i commenti dati alla fine di ogni fase.

LA TECNICA

Vi è una varietà enorme di movimenti, moltiplicati per le quattro specialità, che richiedono molto tempo e molti smanettamenti del joystick per essere ben appresi: posizioni di guardia, piroette, salti, calci e mosse specifiche per ogni specialità marziale richiedono una certa quantità di tempo anche per essere solo esaminate!

Le sequenze di animazione sono comunque sempre molto fluide e realistiche, con efferti sonori consistenti in credibili rumori di contatto osso-osso, osso-arma o arma-arma, ed una gradevole musichetta orientale di sottofondo.

Per dare un'idea dello sfoggio di tecnica, durante il gioco sullo sfondo volano uccellini e si muovono foglie...

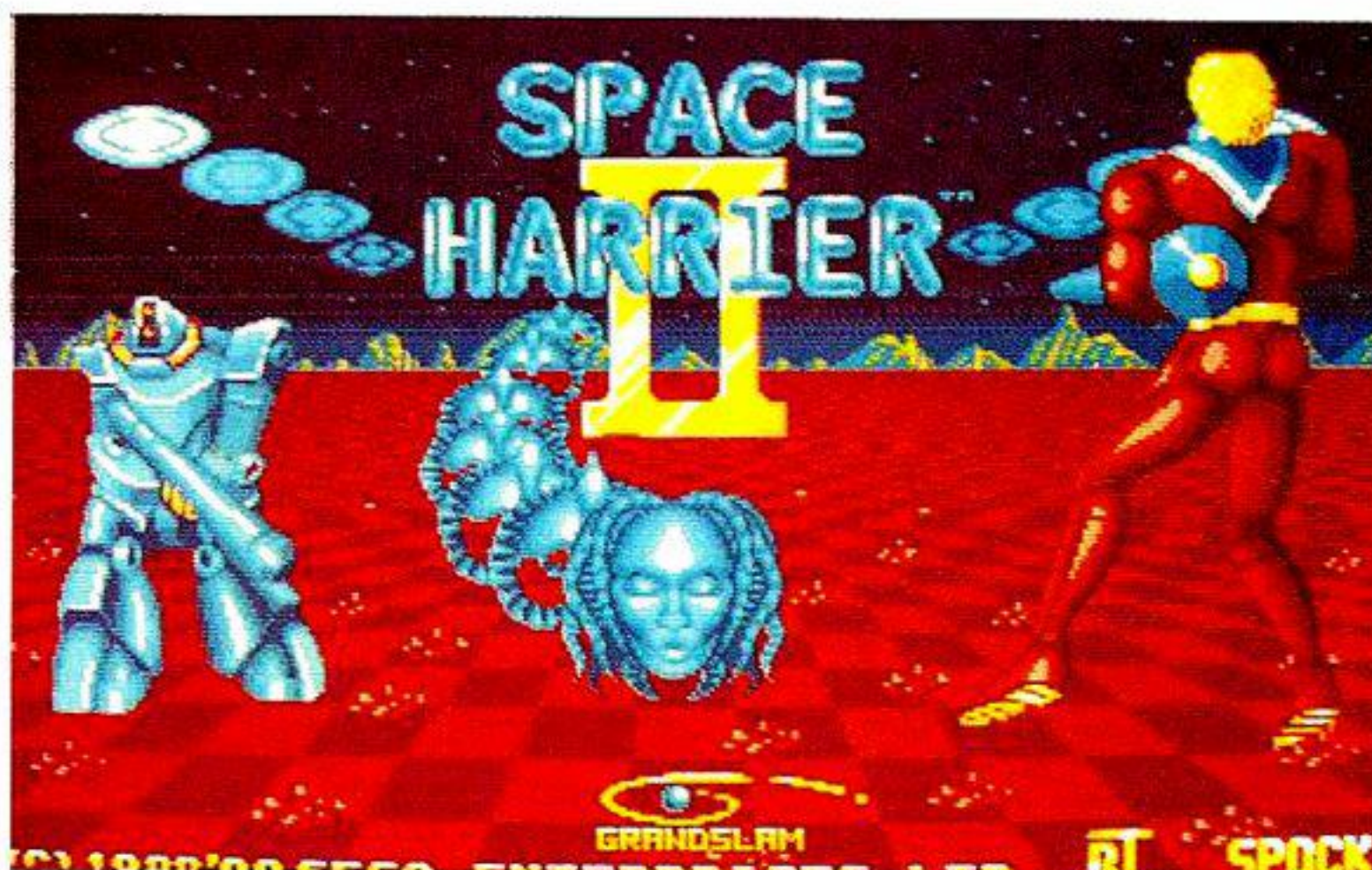
Un chiaro neo è invece l'eccessiva lentezza degli spostamenti ai livelli inferiori, dove anche la scarsa combattività dell'avversario rendono il gioco statico.

IL VOTO

Un bel programma, originale, molto vario e realizzato benino. 7 1/2.

SPACE HARRIER II

Se avete nervi saldi, provate a pilotare con precisione il veicolo a vostra disposizione

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Arcade Game
Softhouse: Sega

Space Harrier è uno di quei giochi classici che non hanno bisogno di presentazioni, anzi sono un termine di riferimento per tutti gli altri videogames che li copiano più o meno sfacciatamente.In pratica si controlla un uomo volante (supposto essere in piedi su di una specie di hovercraft) visto di spalle, armato di *bazooka*, che deve affrontare ad altissima velocità un mondo in prospettiva tridimensionale, velocissimo e pieno di insidie.

Ovviamente vi sono parecchi livelli da affrontare.

Inizialmente si può scegliere se avere il fuoco rapido (continuato) oppure no, se usare il *joystick* oppure il *mouse* per controllare i movimenti, la direzione da cui osservare il movimento, e selezionare effetti sonori o musica di accompagnamento.

Si può scegliere inizialmente, col mouse od il joystick, tra tredici differenti tipi di scenografia, tutti caratterizzati da uno sfondo scorrevole ad alta velocità.

LA TECNICA

Il succo di questo gioco sta nell'estrema velocità di scorrimento dello sfondo e nella nitidezza dei nemici che si materializzano dallo sfondo;questo, dal canto suo, è piuttosto uniforme e banalotto, senza alcuna raffinatezza tecnica degna di nota.

La musica è piuttosto ripetitiva e, certamente, si potrebbe fare di meglio dal punto di vista dell'interfaccia col joystick ed il mouse.

IL VOTO

Un gioco concettualmente banale, ma realizzato piuttosto bene. 7-.

Una nuova simulazione del gioco del tennis in cui il realismo tocca livelli mai visti prima

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Simulazione sportiva
Softhouse: Loriciels

TENNIS CUP

Il gioco del tennis conta ormai parecchie versioni per Amiga, ma questa è sicuramente una delle più rifinite e tecnicamente interessanti.

Si inizia scegliendo col joystick tra gioco singolo contro il computer, gioco a due persone, dimostrativo ed opzioni globali. Queste ultime consentono una serie notevole di personalizzazioni, come ad esempio la possibilità di giocare al meglio in uno, tre o cinque set.

Iniziando si può scegliere tra un giocatore "esistente" od uno creabile a nostro piacimento, in base ad abilità (percentuale di successi) nel portare i colpi: servizio, dritto, rovescio, volee, smash eccetera.

Ovviamente anche noi otterremo un posto in graduatoria in base ai risultati ottenuti con gli altri giocatori (lo scopo del gioco è diventare i migliori). Inoltre, alla fine di ogni partita, si ottengono delle complete statistiche sui colpi portati.

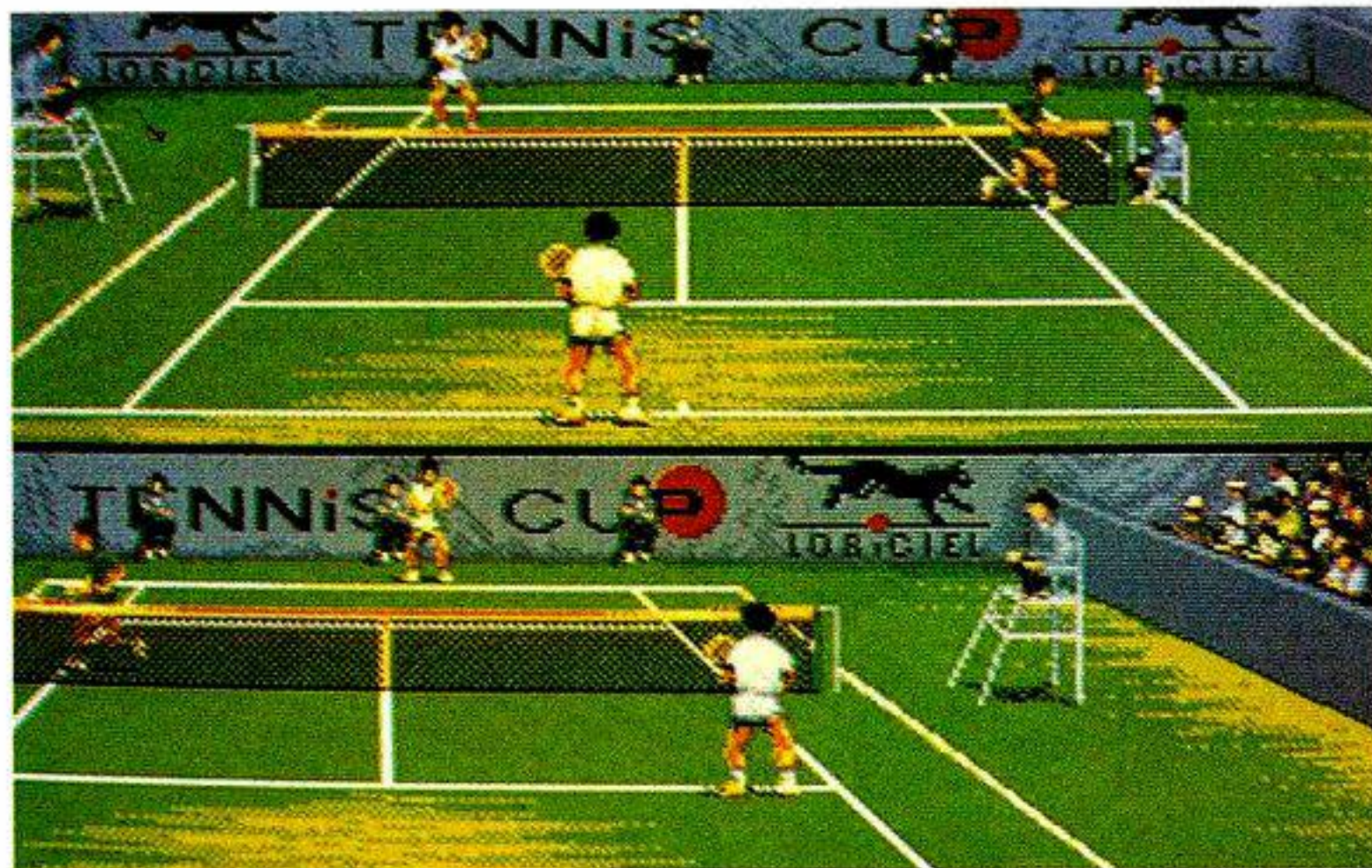
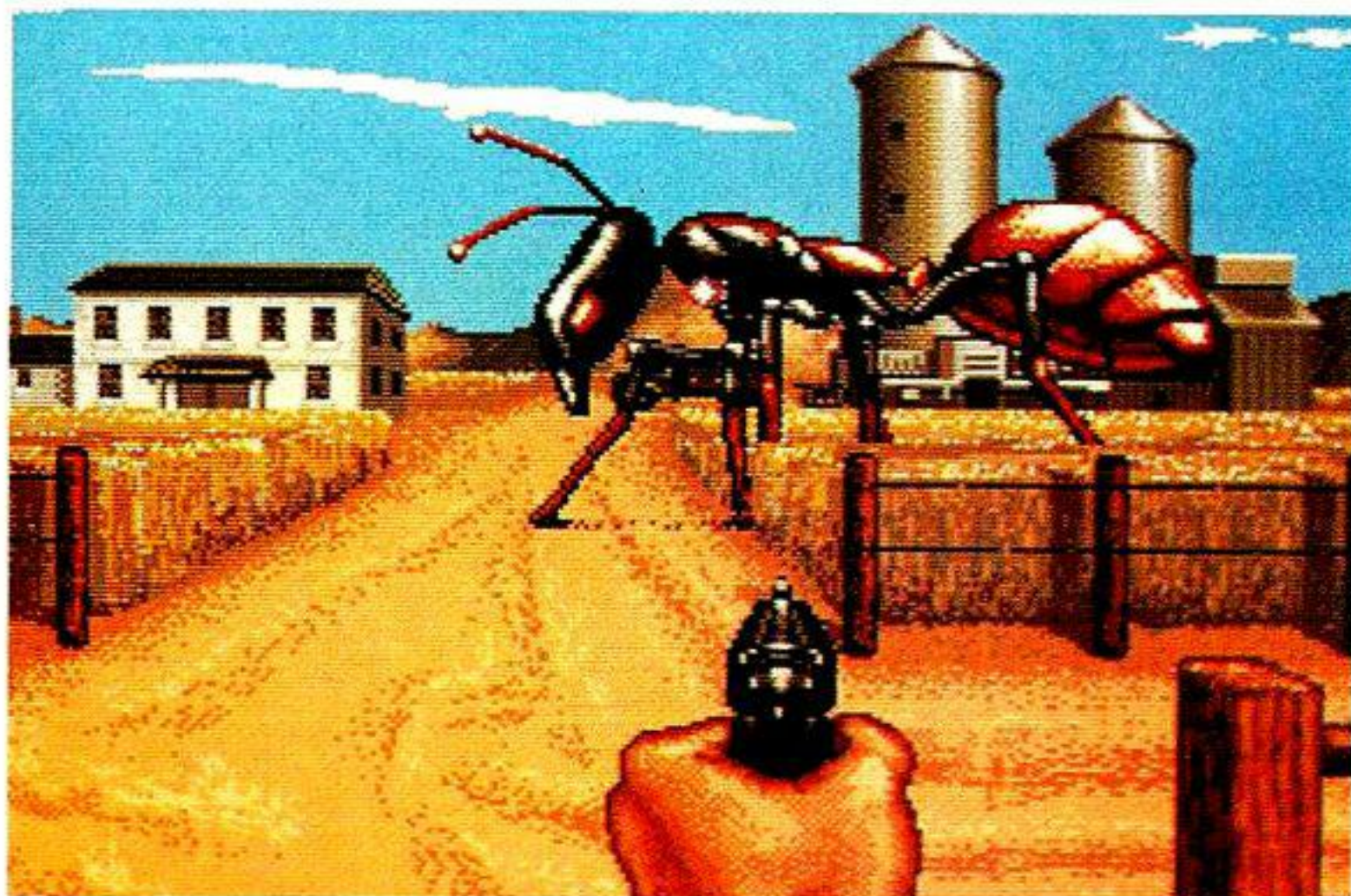
Dopo avere calcolato il grado di abilità, si può indicare se giocare in: esibizione semplice o doppia, allenamento, torneo, coppa Davis o campionato. Ovviamente il livello del gioco cresce proporzionalmente. Come ultima scelta, prima di giocare, si può richiedere un campo in terra battuta, sintetico, in cemento o d'erba (che influiscono sui rimbalzi della pallina).

Il gioco viene condotto muovendo il joystick per portare il giocatore al punto giusto per colpire la palla. Premendo il pulsante di fuoco inizia il movimento e rilasciandolo, eseguendo uno spostamento col joystick, si colpisce la sfera con un particolare effetto. Effettivamente occorre parecchia pratica prima di colpire con una certa costanza la pallina, poi si può iniziare a ricercare gli effetti da assegnare.

LA TECNICA

La scena di gioco è suddivisa in due campi (particolare utile quando giocano due persone), dal momento che ambedue possono guardare la scena di gioco come se fossero posizionati alle spalle del giocatore mosso. Ciò è un grande vantaggio rispetto ad altri giochi dello stesso genere, e se si considera la velocità e la fluidità del movimento, la nitidezza delle sagome e l'entusiasmante qualità delle sintesi sonore, il tutto è veramente interessante.

IL VOTO

Grande varietà di opzioni; tecnica e realismo ineccepibili. 9.

IT CAME FROM THE DESERT

Tanti bei formiconi giganti, da uccidere al più presto, sono i nostri pericolosi nemici

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Avventura grafica
Softhouse: Cinemaware

L'ennesima avventura con animazioni grafiche di altissima qualità dalla *Cinemaware*, sulla scia del film americano *It Came From The Outer Space*, degno erede di *Defender of The Crown, Sinbad, Rocket Ranger, Capone*...

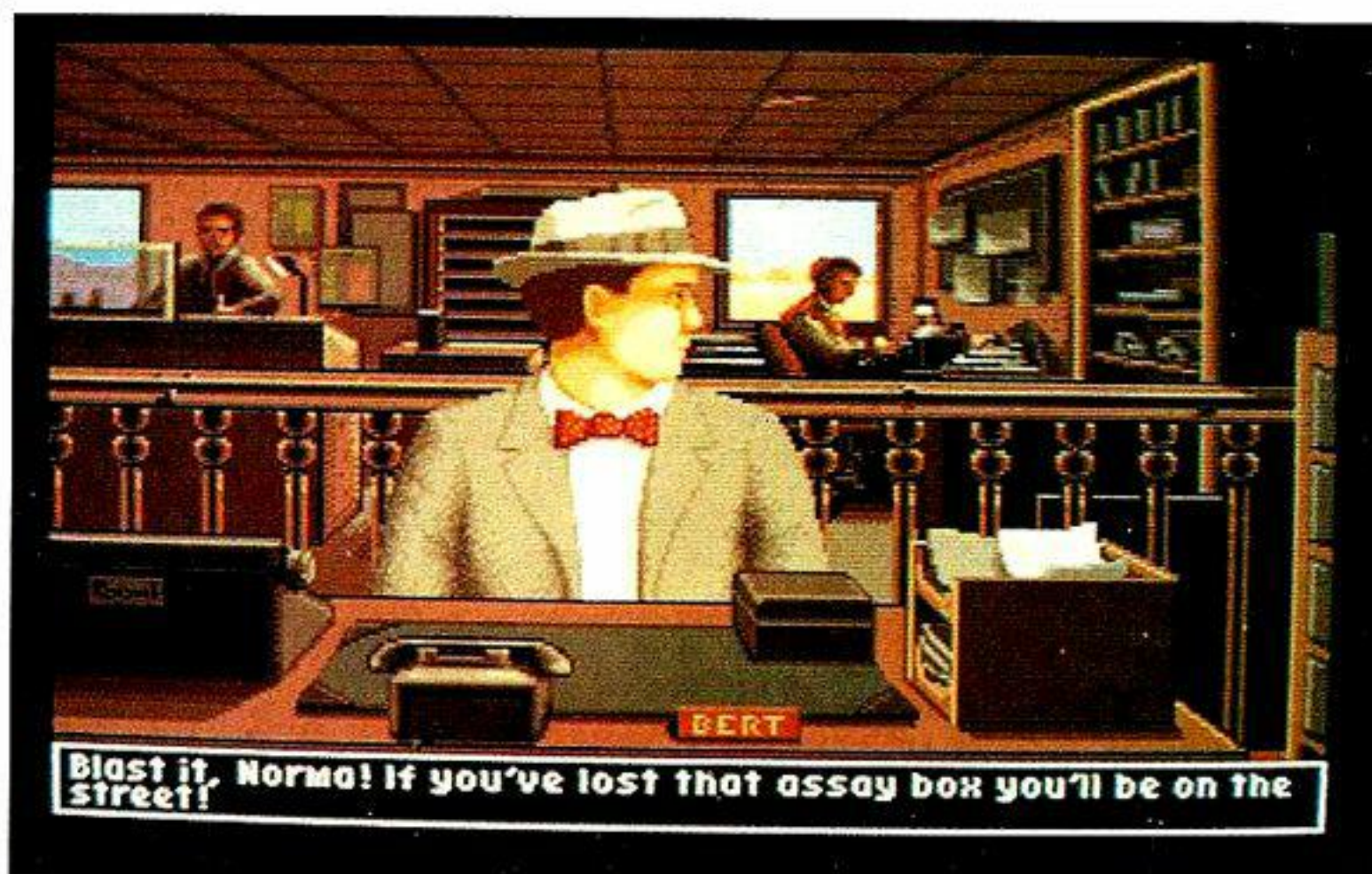
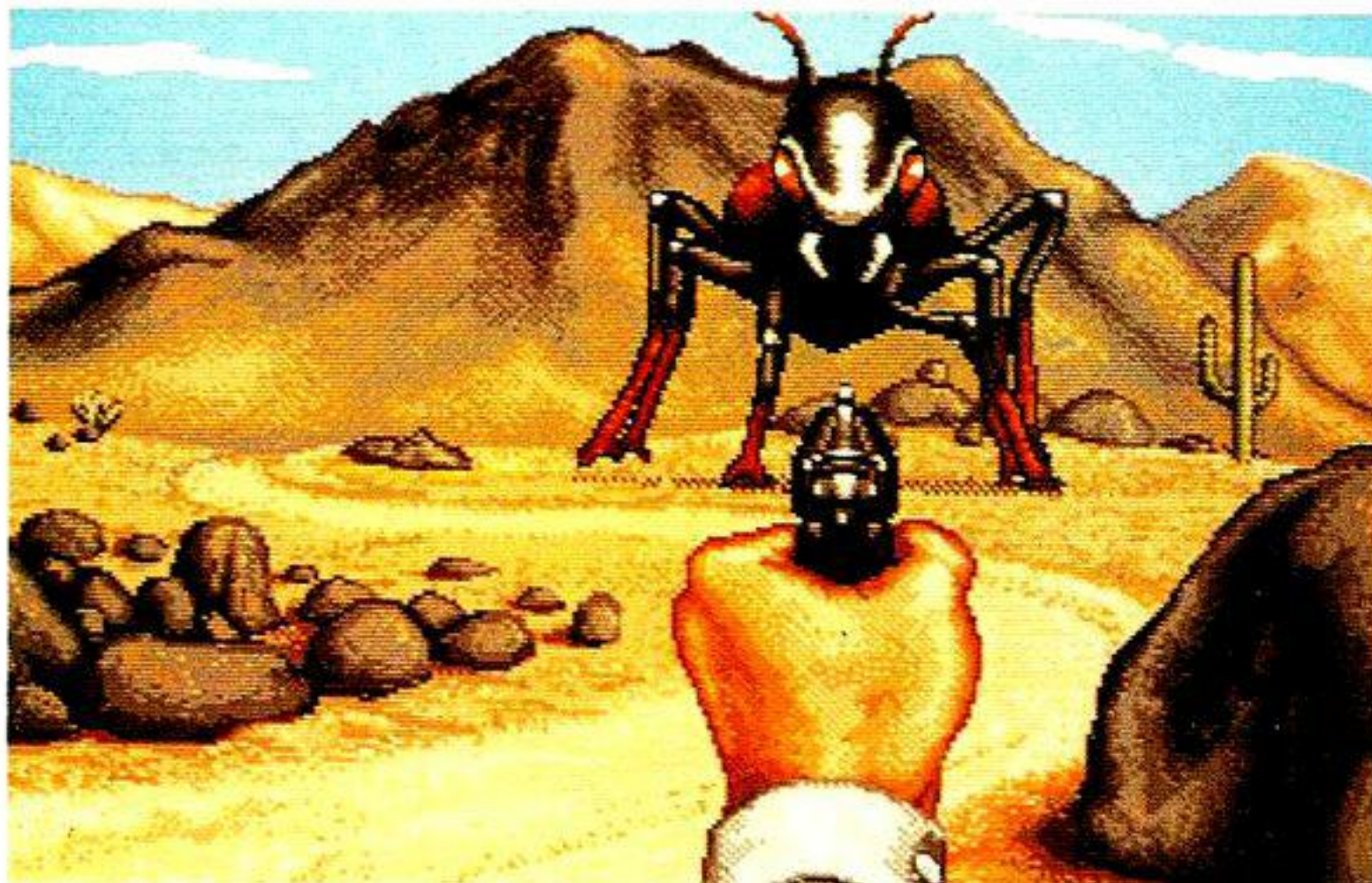
IL GIOCO

Un meteorite è precipitato nel deserto, proprio vicino al nostro amato villaggio, *Lizard Breath*, provocando una mutazione nelle formiche. Vestiamo i panni di un geologo che deve cercare un rimedio entro quindici giorni (tenendo conto che un minuto di gioco è un secondo di tempo, e che bisogna anche dormire), data entro la quale le formicone decideranno certamente di banchettare nel villaggio.

Fondamentalmente, per riuscire a convincere il sindaco a darvi retta, bisogna raccogliere quattro prove fondamentali (come e quali a voi scoprirlo), al che egli acconsentirà alla fine di distribuire operai delle miniere, contadini, polizia ed esercito per uno scontro finale risolutivo con i formiconi (nome scientifico *Pugonomyrex Rugosus*).

Come è consuetudine dei prodotti Cinemaware (da notare che il programma è stato scritto dalla *Mirrorsoft*), il gioco è effettivamente un misto di avventura pilotata da frasi scelte da menu (con il joystick) e fasi arcade vere e proprie. Queste consistono in: *fughe* da ospedali, *risse* all'arma bianca, *disinfestazione* da aereoplano, *incidenti* stradali, *lotta* coi formiconi a suon di pistola ed altro.

Alcune di queste fasi sono veramente notevoli: la fuga dall'ospedale, ad esempio, potrebbe quasi essere un gioco a se stante (in stile vagamente *Gauntlet)* con una grafica dall'alto piccolissima ma dettagliatissima (quando ci si nasconde a letto le coperte salgono fin sulla faccia) ed il trucco sta nel camminare dietro le infermiere per non farsi vedere.

L'avventura, invece, corre per binari tutti da scoprire. Ad esempio il punto debole dei formiconi, cioè dove mirare con la pistola durante la fase arcade, viene rivelato dallo scienziato del laboratorio di Lizard Breath. Analogamente, fare indagare su strane pietre radioattive dal nostro assistente potrebbe fare bruciare la nostra casa (se non si è bravi con

l'estintore), mentre dare troppa corda ad una bella fanciulla che ha avuto un incidente e chiede soccorso potrebbe causare più grane di quanto si potrebbe credere.

La città comprende vari luoghi, tutti dotati della classica scenografia a quadretto della Cinemaware: pub, radio privata, miniere, laboratori, tribunali, redazione, strade, aereoporto, fattorie, cinema (proiettano *Rocket Ranger*, guarda caso), studio di una chiromante (che ricorderà la zingara di *Sinbad* a più di un lettore), Ospedale e via girovagando. Per così tanti luoghi esistono ovviamente un buon numero di personaggi dalle personalità più o meno spiccate, da capire per scoprire come farli collaborare con noi.

LA TECNICA

Il gioco è distribuito su tre dischetti, dei quali il primo contiene l'introduzione tipo "film" ed il programma vero e proprio, mentre gli altri due dischi comprendono tutte le scene e le sequenze di animazione. In pratica è quindi indispensabile, per chi non possiede un disco rigido, disporre di almeno due drive, dove dopo l'avviamento lasceremo sempre inseriti il secondo ed il terzo disco. I tempi di caricamento non sono comunque mai lunghissimi.

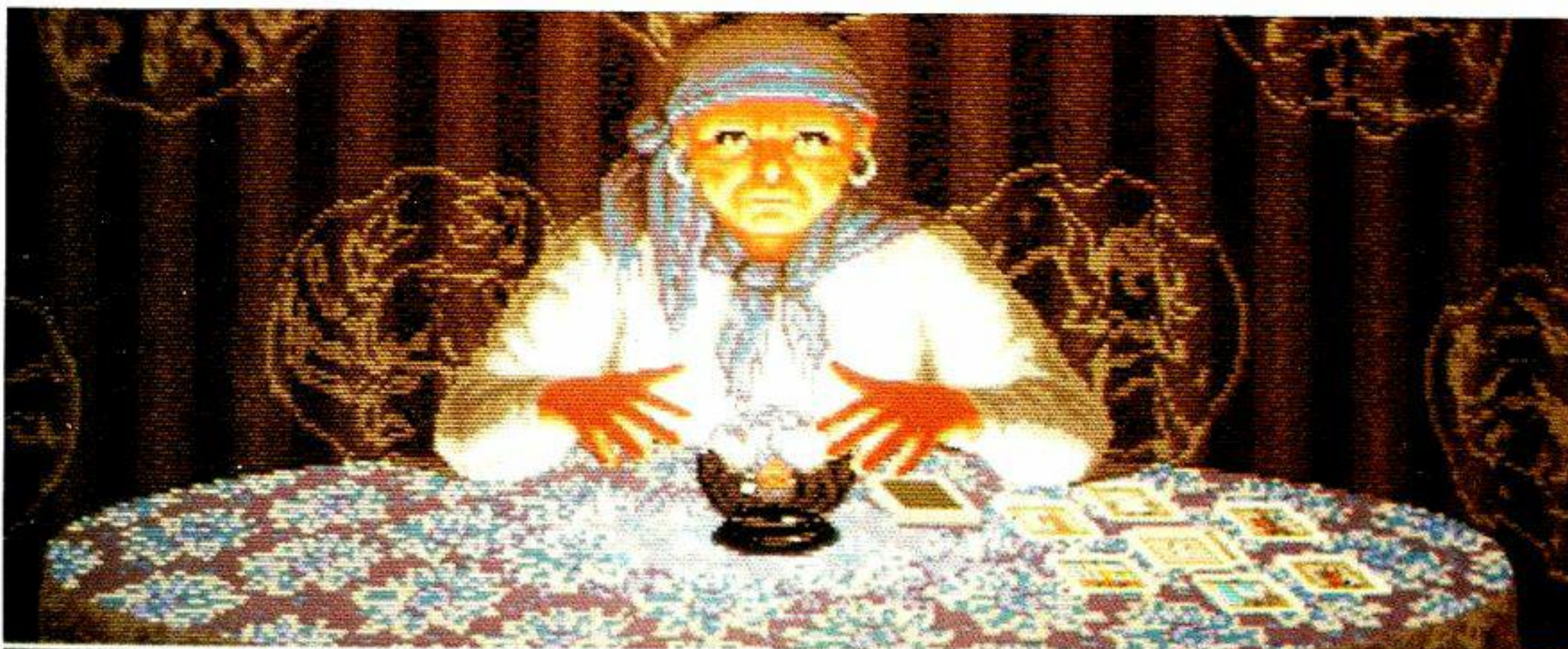
La grafica è nella migliore tradizione Cinemaware: il sapore anni '50 pervade tutte le scene di gioco e si nota una cura maniacale in particolari e dettagli che si scoprono sempre nuovi anche dopo parecchie ore di gioco: il trucco e le acconciature delle donne, le automobili, la segnaletica stradale, l'architettura dei palazzi e tutto il resto.

Gli effetti sonori consistono in una angosciante musica di accompagnamento, con alcune variazioni, e numerosi suoni campionati per le varie scene arcade: spari, urla, spiaccicamento di mostri, volo d'aeroplano e via digitalizzando.

L'interfaccia tramite joystick (per scegliere le frasi e le azioni dal menu scorrevole che compare dal basso) è rapida ed efficace.

Da notare che quando compare il menu, una scelta considerata "più ovvia" è già selezionata di default e basta semplicemente premere il pulsante di fuoco per procedere.

IL VOTO

Un classico. Considerata la trama appassionante, la varietà di possibili soluzioni e trame, la qualità tecnica e la buona interattività diamo di cuore un bel 9.

SIM CITY

Finalmente un gioco didattico ed educativo insieme: come "funziona" una città

Computer: Amiga (1 Megabyte)
Gestione: Mouse
Tipo: Strategico
Softhouse: Infogrames

Un gioco stile "Populous", dove però si controlla la nascita e l'evoluzione di una città con criteri statistici e scientifici.

IL GIOCO

La simulazione della vita di una città si svolge su di una quantità di fattori interdipendenti: densità di popolazione, inquinamento, mezzi di trasporto, disponibilità di servizi e di posti di lavoro, protezione della Polizia contro i malviventi, fornitura di energia elettrica, percentuale di parchi e di zone pesantemente industrializzate, protezione civile da parte dei vigili del fuoco, capacità di scambiare merci per mezzo di aerei, porti, eccetera, eccetera.

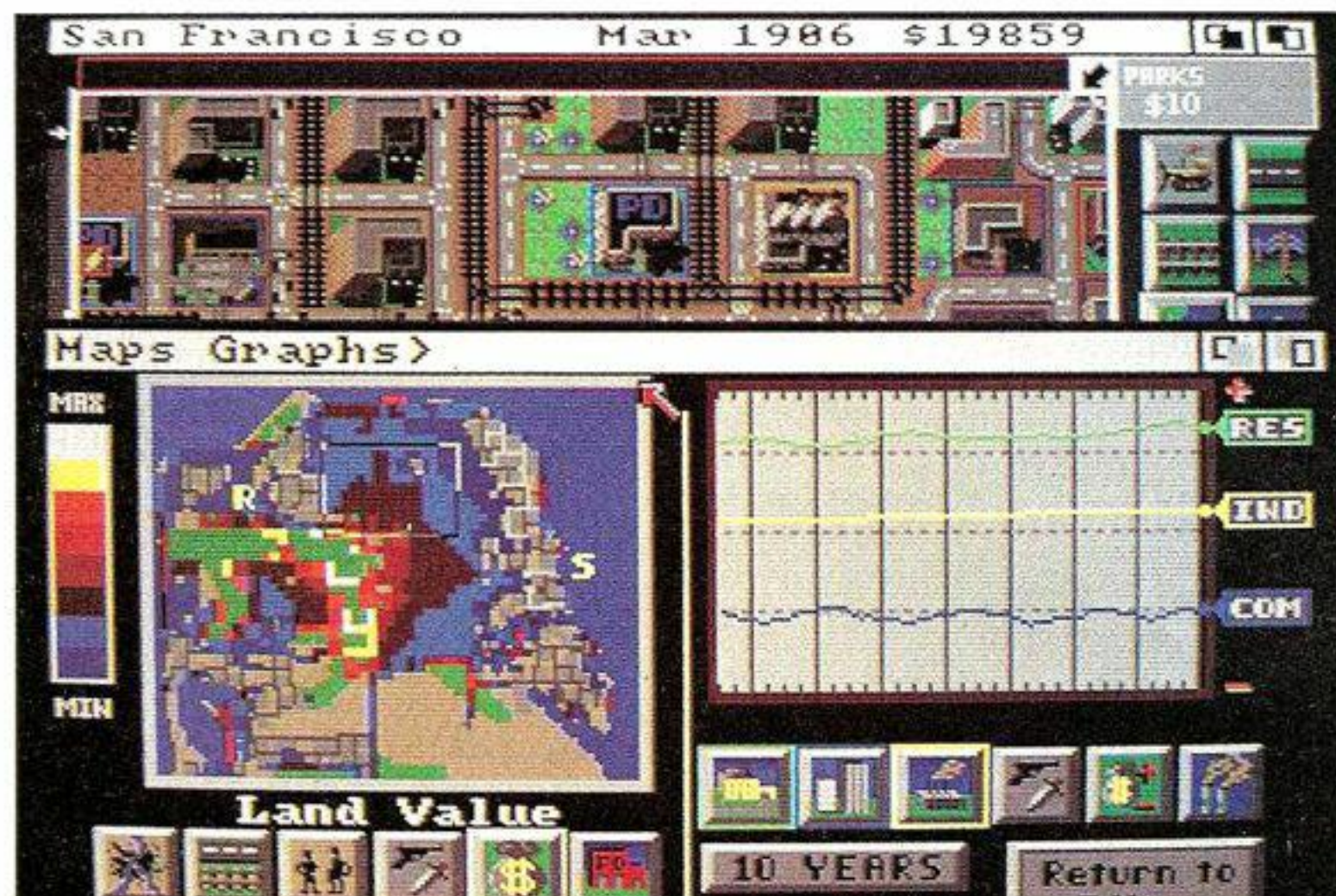
Partendo dal nulla, solo con una somma in dollari proporzionale al livello di gioco scelto, il giocatore deve creare centri residenziali, commerciali ed industriali, scegliere la collocazione delle centrali elettriche e delle relative linee di collegamento tra i vari centri, collocare il verde pubblico e le centrali di polizia per combattere la criminalità e, insomma, tenere in equilibrio centinaia di fattori interdipendendti tra loro.

Ad esempio, inizialmente bisogna fornire alla città più case (che non uffici o industrie) per favorire l'insediamento di cittadini, ma col proseguire del tempo tale equilibrio dovrà essere invertito.

Si tenga presente, però, che troppe industrie favoriscono l'inquinamento, e quando invece si hanno troppi cittadini ed un'inadeguata rete di trasporto (ferrovie o strade) si forma troppo traffico (visibile) e il commercio scarseggia.

Inoltre pochi cittadini forniscono soldi pagando le tasse: se si alza la percentuale di tassazione (da menu), si guadagna di più, ma i cittadini tenderanno ad emigrare ed il commercio e l'industria ne risentiranno a medio termine.

Molti cittadini e zone con scarso verde facilitano la criminalità, che scoraggia gli insediamenti e costringe ad impiantare dipartimenti di Polizia dappertutto, che poi sottraggono buone fette degli introiti delle tasse.

Si potrebbe proseguire nella descrizione degli innumerevoli fattori di equilibrio di Sim City, un programma che ha riscosso tanto successo da essere stato prodotto praticamente per qualunque modello di computer (perfino per i seriosi

MS/DOS, PS/2 e Macintosh). Non per niente il programma è fornito con un ponderoso manualetto di una cinquantia di pagine, ricco anche di consigli e di una bibliografia per erudirsi sul come funziona una città "ideale".

Sono anche previste alcune scene realistiche ed inventate (anche con mostri tipo *King Kong* che spaccano tutto) per potere offrire nuove situazioni di gioco a chi è già diventato esperto.

LA TECNICA

La versione Amiga sfrutta bene l'enorme facilità di interfacciamento prevista da *Intuition* con una quantità di gadget, finestre e menu.

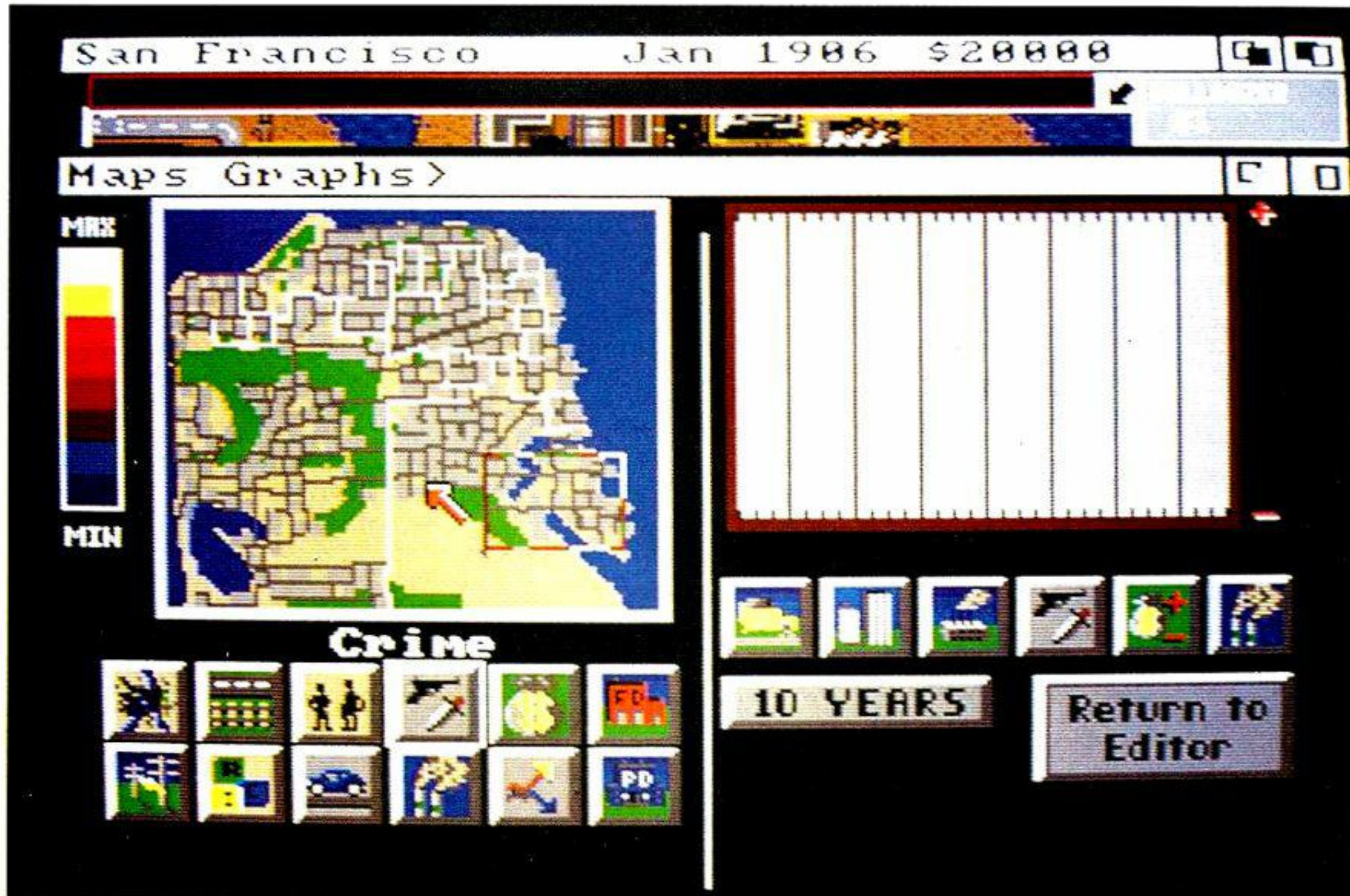
Il grado di dettagli è notevole: trenini, automobili, fontane, elicotteri, tifosi nello stadio, aeroplani, navi, ponti sono beno mossi nelle loro piccole dimensioni. I gadget e le schermate con i grafici sono realizzate con cura e dovizia di particolari.

L'interfacciamento via mouse è eccellente, alternativamente si possono usare combinazioni di tasti.

Gli effetti sonori sono pochi ma davvero buoni.

Il voto

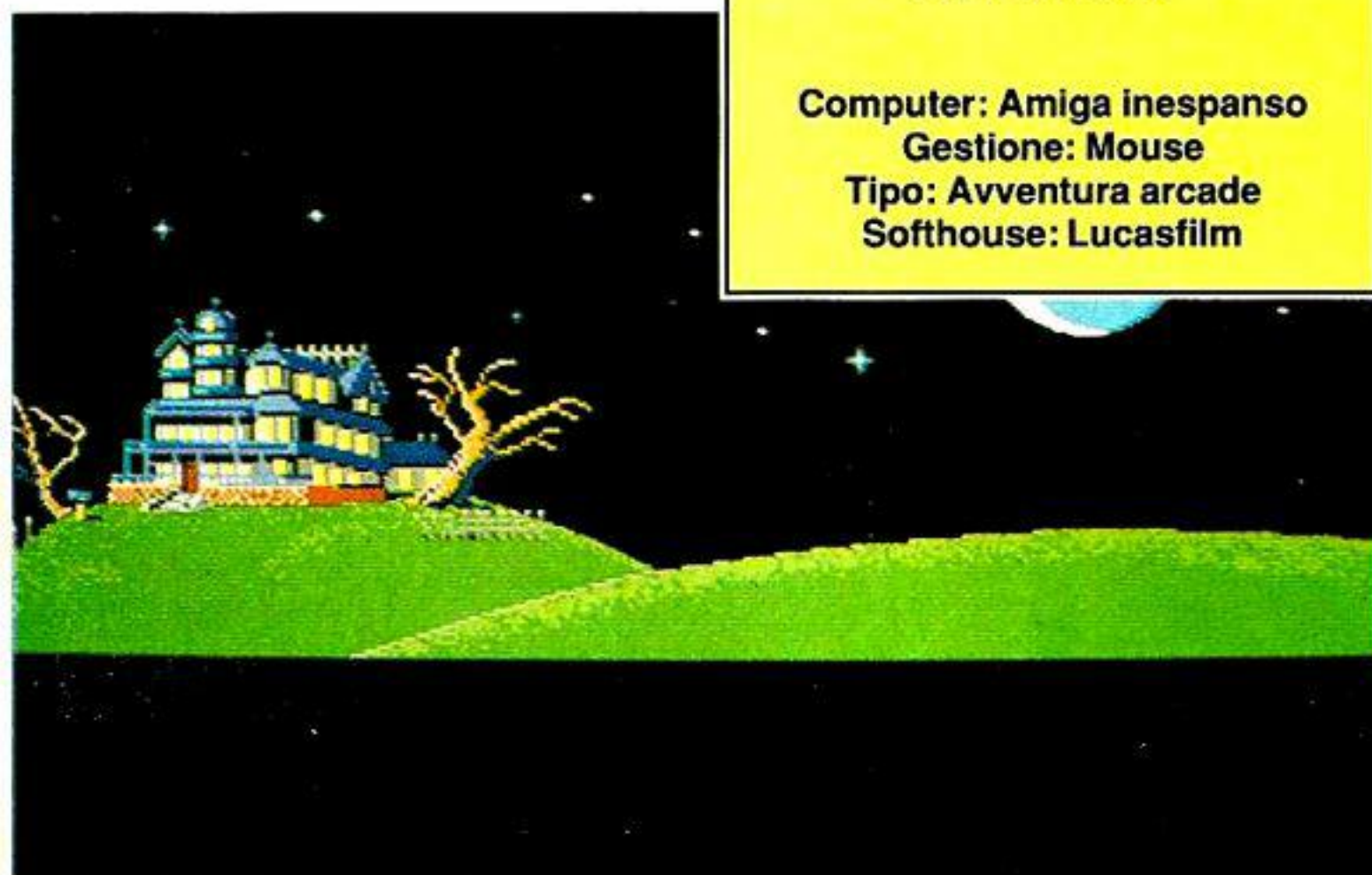
Un gioco originale, scientifico, educativo, vario, con incentivi a volontà, adatto a chiunque. 9.

MANIAC MANSION

Una bella avventura alla *Indiana Jones* e *Zak McKracken*, estremamente divertente e con numerosissimi simpatici problemi da risolvere, fanno di questo programma un best seller internazionale.

In aiuto della nostra fidanzata, con qualche pericolo da evitare, ma con humor

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Avventura arcade
Softhouse: Lucasfilm

IL GIOCO

Una meteora si è schiantata nei pressi della casa dello scienziato *Weird Ed*. L'oggetto celeste è semplicemente un'entità aliena, che fa impazzire il dottore e gli mette in testa strane idee su come diventare padrone del mondo. Egli costruisce, infatti, una macchina per impadronirsi di posti importanti, che funziona rubando l'intelligenza alle persone. Prima di avviarsi alla prova di lavoro, egli però desidera sperimentarla; quindi rapisce *Cindy*, studentessa dell'attiguo college. Come tutte le eroine da videogioco, Cindy è fidanzata, con Dave, che lotta come un pazzo (lo controlliamo noi) per liberarla. In virtù della massima "chi trova un amico trova un tesoro", inizialmente si possono scegliere due amici tra gli studenti del College per farsi aiutare: *Syd* (cantante New Wave), *Wendy* (aspirante giornalista), *Bernard* (cervellone del corso di Fisica), *Jeff* (atleta), *Michael* (fotografo) e *Razor* (cantante punk).

Il bello del gioco è che la trama e le istruzioni allegate non dicono ovviamente granchè della soluzione. Anzi, solo chi possiede il *pacchetto originale* può leggere su di un foglio specialissimo (non fotocopiabile in alcun modo, ed assai dispendioso da ricopiare) alcuni suggerimenti fondamentali per sapere come si procede nel gioco in taluni punti critici, altrimenti del tutto incomprensibili.

Un deciso senso di "humor" riveste moltissime delle situazioni di gioco che si incontrano, con particolari e colpi di scena veramente da barzelletta. Ovviamente ogni personaggio ha una propria personalità e può, almeno teoricamente, completare l'avventura.

Ciò contribuisce a moltiplicare il già alto numero di possibili modi di soluzione, che contribuiscono a mantenere l'interesse anche quando si è risolto il gioco.

LA TECNICA

L'interfaccia col numero limitato di frasi fatte selezionabili col mouse, col quale si possono anche indicare direttamente gli oggetti interessati alle azioni sullo schermo, è certamente considerato il migliore attualmente in circolazione, soprattuto dal punto di vista di chi non conosce perfettamente l'inglese. Infatti è sufficiente imparare il significato dei verbi sullo schermo; poi si può giocare, a patto di avere il foglio infotocopiabile.

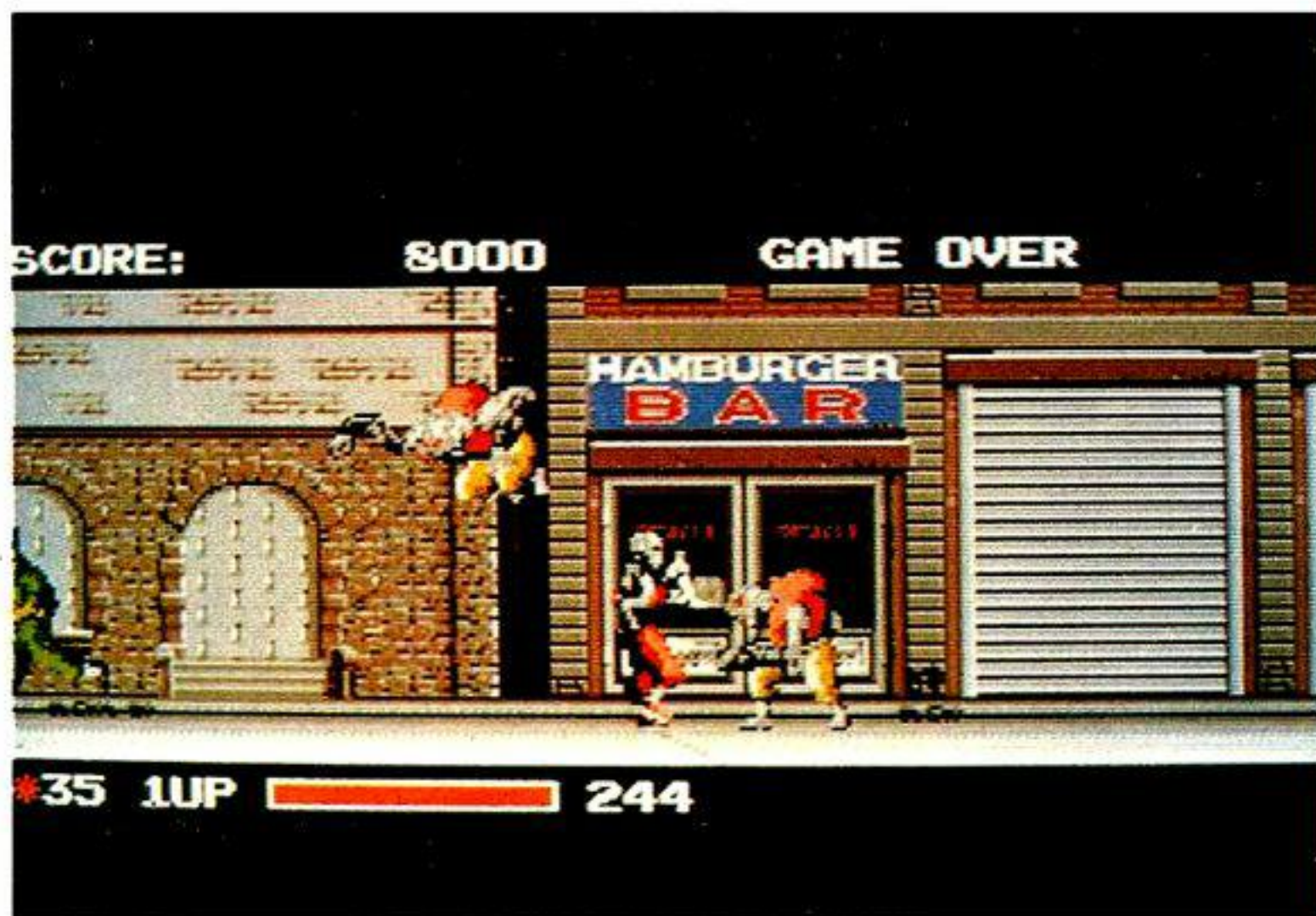
La grafica è in animazione decisamente buona, gli sprites sono belli alti, comicamente sproporzionati e seguono docilmente i comandi impartiti dal mouse. Gli effetti sonori sono semplici ma divertenti.

IL VOTO

Un classico per Amiga, nel suo genere e non, consigliato a tutti. 9.

NINJA WARRIORS

Abilità e destrezza sono tassative in questa conversione da un noto coin up

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade combattimento
Softhouse: Virgin

Raramente si può definire perfetta la conversione di un *top hit*; questo programma per Amiga, però, si avvicina molto all'ideale, considerate le ovvie differenze tecnologiche tra un computer *dedicato* ed un *personal* computer.

IL GIOCO

Con il celebre *Silkworm* dell'anno scorso, *Ninja Warriors* ha due buoni punti di contatto: è stato convertito da due eccellenti programmatori, (Ronald Pieket Weeserik e John Crudy), che hanno il vizietto di realizzare conversioni migliori dell'originale sfruttando eventuali caratteristiche speciali del computer ospite. In questo caso era comunque davvero difficile restare perfettamente fedeli, in quanto la versione di sala giochi è famosa per il triplo monitor e tristemente nota per la pochezza della trama.

Ninja Warriors per Amiga risulta quindi un gioco che gli inglesi definiscono *Two Player Horizontal Beat Em Up*, ovvero "spaccatutto a due giocatori con scorrimento orizzontale".

Sono previsti sei livelli di gioco, con il più largo ampio circa diciassette volte lo schermo. Si incontrano strade, aereoporti, interni di case e tanto altro ancora. Inizialmente si è armati solo di un limitato numero di *Shuriken* (stelle appuntite) e si procede incontrando una varietà di nemici: esseri con più gambe e braccia che denti, esseri con più denti che peli, esseri con più peli di Lucio Dalla eccetera. Tutti comunque con la caratteristica di essere ottimi pestatori, guarda caso.

LA TECNICA

La grafica è fedele all'originale, comprimendola per farla brillare senza problemi anche su un solo monitor, anzi qualcuno sostiene che sia stata trasferita direttamente con un *Download* dalle ROM del videogioco originale. Si consideri che i lunghissimi sfondi non si ripetono mai e non sono generati algoritmicamente.

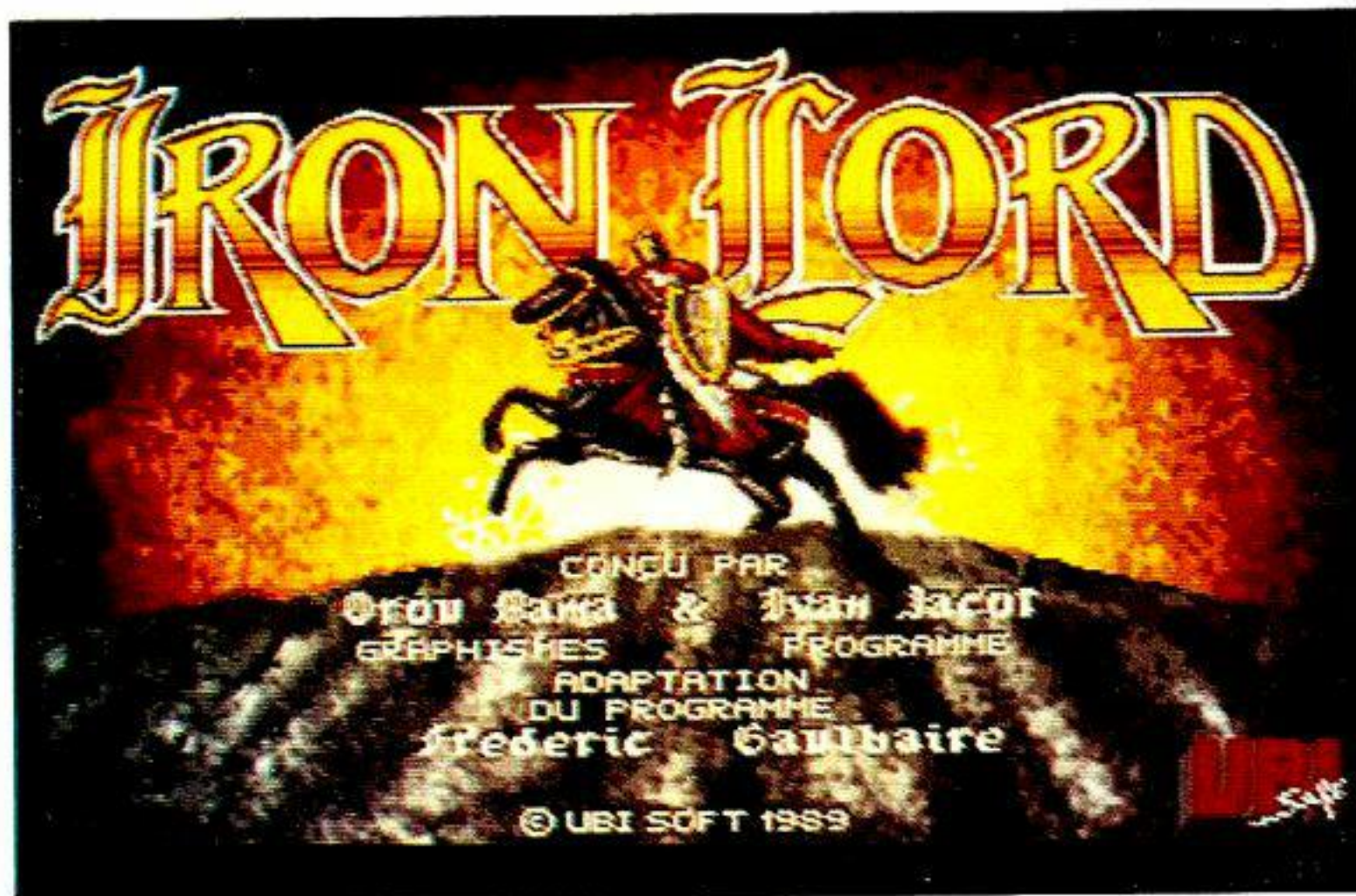
Comunque, nonostante l'area di gioco sia una striscia piuttosto stretta rispetto all'originale, i ninja sono pur sempre alti circa sei sprites ed animati splendidamente; suggeriamo di ammirare in particolare l'ondeggiamento dei capelli durante i salti e l'animazione del carrarmato, consistente in ben diciassette fotogrammi. Gli effetti sonori sono all'altezza

IL VOTO

Praticamente indistinguibile dal classico videogioco originale, è un picchiaduro di pregevole fattura. 8 e mezzo.

IRON LORD

Un misto di Arcade e Adventure, per nulla all'altezza di prodotti già noti da tempo

Computer: Amiga inesp. C/64
Gestione: Joystick, Mouse
Tipo: Arcade/strategico
Softhouse: Ubi Soft

Si tratta di un programma arcade e strategico stile *Cinemaware*, che ha richiesto circa quindici mesi per essere completato ed è distribuito in tre lingue: inglese, francese e tedesco.

IL GIOCO

Destinati ad ereditare il trono, quali principi ereditari, abbiamo avuto la sfortuna di imparentarci con uno zio che ha assassinato gli altri membri della famiglia ed ha distrutto il nostro castello.

Non contento, lo zietto ha radunato un esercito diabolico e si accinge ad attaccare i nostri possedimenti.

Bisogna dunque abbandonare le mura distrutte del castello ed ispezionare i paesi circostanti.

Un rettangolo nella porzione superiore destra dello schermo presenta la visione dall'alto di paesi e luoghi, mentre quando si incontra qualche essere vivente compare un menu che consente di scegliere tra alcune operazioni banali: dare, parlare, acquistare eccetera.

La grafica è sempre molto curata: personaggi che camminano, mappe decorate "a mano" in perfetto stile medievale, pittoreschi personaggi animati, eccetera.

Come nei giochi della Cinemaware, durante il gioco si può accedere a varie fasi più puramente "arcade", tipo un *tiro* con l'arco in stile simile a quello dei giochi sportivi (a parte i vestiti e le armature del tiratore, invece delle tutine in fibra sintetica), il *braccio di ferro*, la *scherma*, il gioco dei *dadi* (puramente casuale) ed altro ancora. Superando le varie fasi arcade si acquista prestigio e riesce più facile radunare l'esercito da schierare contro lo zio cattivo. Altro metodo per assemblare un buon esercito consiste nello svolgere (ruffianamente) missioni per potenti.

LA TECNICA

Si tratta del classico prodotto basato molto più sull'impatto iniziale nei confronti dell'acquirente che sui reali contenuti ludici.

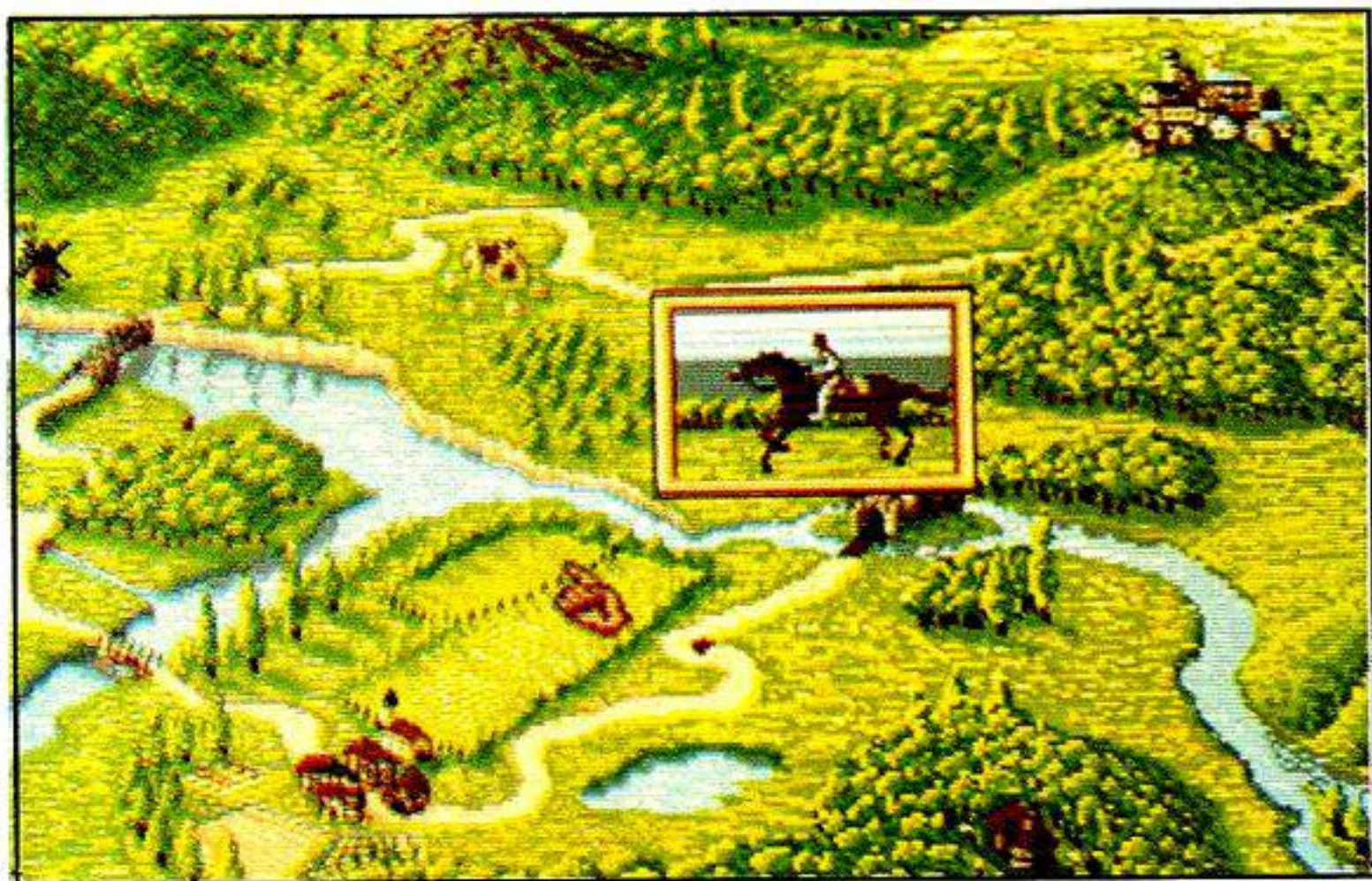
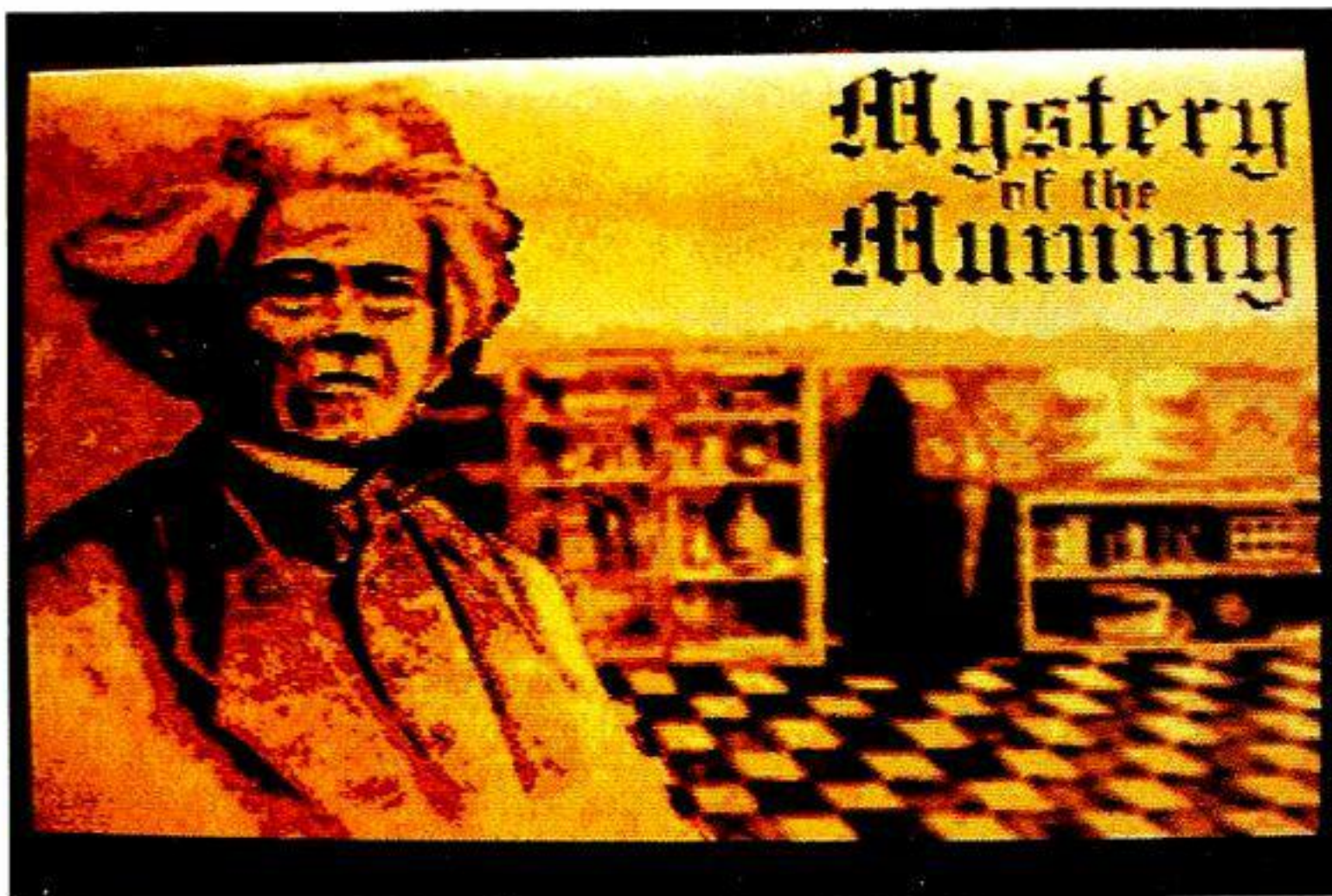
Ciò significa che la grafica di scena e di movimento e gli effetti sonori sono curati con grande perizia, ma sono del tutto carenti incentivi, innovazioni ed originalità.

IL VOTO

Solo per i patiti del genere Cinemaware "impoverito". 6+.

MYSTERY OF THE MUMMY

Avete mai avuto a che fare con una mummia? Non abbiate paura, è piuttosto morta...

Computer: Amiga inespanso
Gestione: Mouse, tastiera
Tipo: Avventura
Softhouse: Rainbow Arts

Ecco un'avventura nei panni di un investigatore agli inizi del secolo, per gli appassionati più esigenti.

IL GIOCO

Vestiamo i panni di un avvocato di Amburgo che riceve la richiesta di aiuto da parte di un onesto cittadino, *Rudolf Rabensberg*, che ha perso una mummia appartenente al padre. Sapendo che ad un'asta è stata messa in palio una mummia vagamente familiare, Rudolf ci chiede di indagare per scoprire se si tratta dell'eredità paterna rubata in passato. La scena del gioco si suddivide tra Amburgo e Monaco, dove si svolgono le indagini, ambientate nell'anno 1912.

Inizialmente il gioco consiste soprattutto nell'indagine in vari posti misteriosi e ricchi di indizi sparsi per la città. Si consideri che esistono più di un centinaio di personaggi, molti dei quali sono totalmente sconosciuti e non citati nelle istruzioni, per cui bisogna arrangiarsi a chiedere ad uno e all'altro per procedere.

Oltre alle scene di riflessione su più elementi ed indizi contemporaneamente, sono previste fasi di azione pura, come ad esempio l'improvvisarsi farmacisti in un laboratorio chimico, o subacquei per recuperare reperti in fondo ad un lago.

LA TECNICA

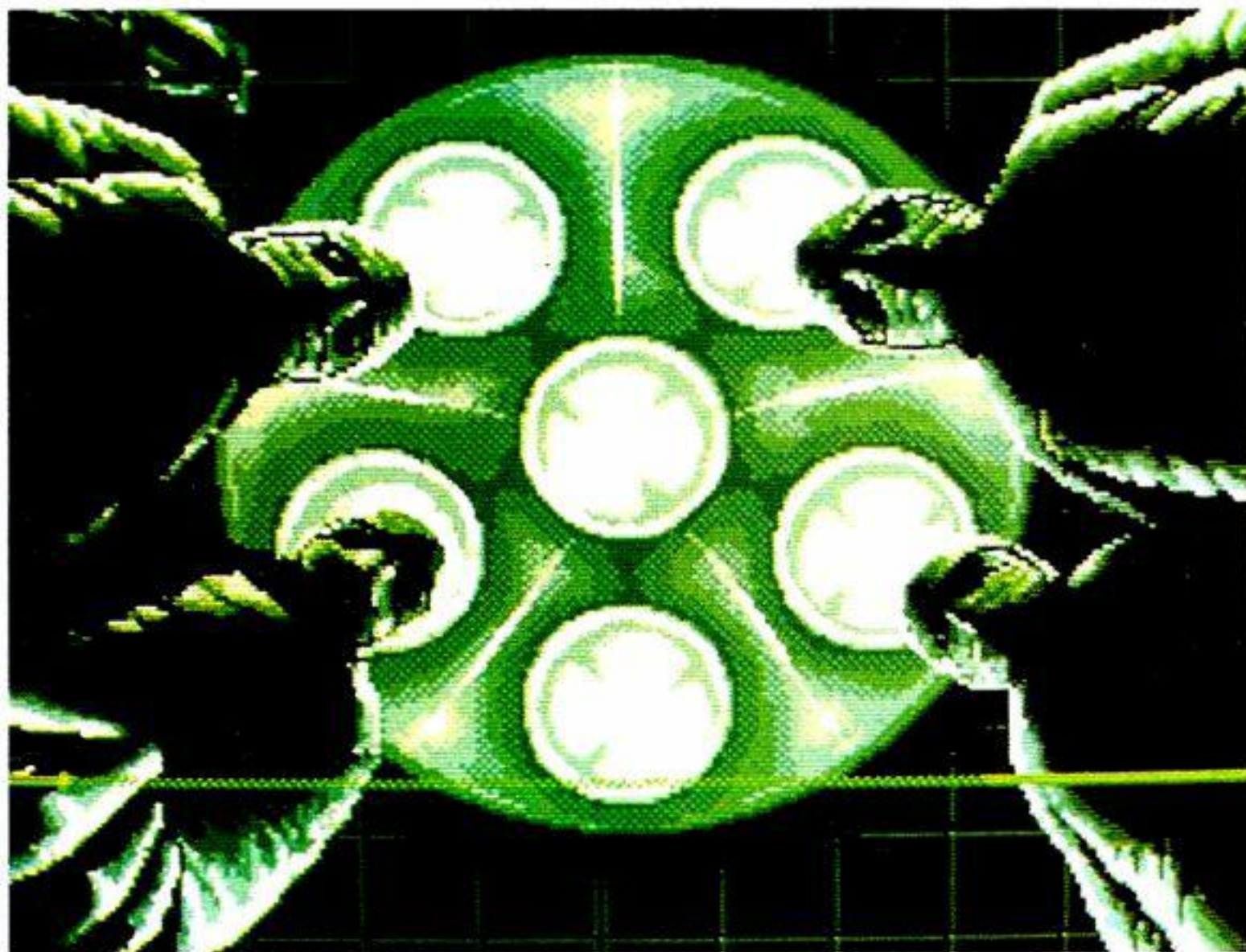
Lo schermo è suddiviso in varie sezioni e col mouse, clickando opportunamente su particolari delle schermate grafiche, si aprono varie finestre che consentono di esaminare particolari ed indizi.

Da tenere d'occhio anche i quattro indicatori grafici di *fame, stanchezza, soldi* e *sete*.

L'interfacciamento con l'utente avviene tramite mouse e gadget sullo schermo, oppure digitando su tastiera eventuali nomi o stringhe alfabetiche.

IL VOTO

Una discreta avventura, con interfaccia utente originale (anche se ridotta) molti incentivi ma poca varietà di situazioni. 6 e mezzo.

WEIRD DREAMS

Un'operazione chirurgica mal riuscita ci proietta in un mondo di incubi

Computer: Amiga inespanso
Gestione: Joystick
Tipo: arcade multifase
Softhouse: Rainbird

I concetti alla base della trama di *Weird Dreams* sono da ricercare nelle teorie del subconscio e dell'immaginazione umana.

Mentre siamo ricoverati in sala operatoria per una semplice operazione (tonsille?) andiamo in coma.

Ovviamente ciò non inibisce tutte le funzioni vitali e, dalla visione dei camici bianchi piegati su di noi, passiamo allo sfrenato mondo dei sogni, anzi degli incubi.

IL GIOCO

Il gioco vero e proprio consta di più fasi differenti, col nostro protagonista rappresentato da una figura stralunata, perennemente in pigiama.

Nella prima fase, ad esempio, siamo all'interno di una macchina per produrre lo zucchero filato (o è la macchina ad essere grossa, o siamo noi ad essere piccoli o è proprio un incubo!) con un'asta centrale che rotea e minaccia di colpirci con oggetti appiccicosi. Anche altre fasi sono ambientate chiaramente in un *Luna Park*, almeno come sfondo, dopo essere passati per una rassicurante (si fa per dire) "camera degli specchi".

LA TECNICA

La grafica di sfondo è sicuramente ad ottimi livelli, mentre quella di animazione risente ancora di una tecnica che non sfrutta al massimo le caratteristiche Amiga. In effetti il gioco è sicuramente un pò (troppo) lento, anche tralasciando le fasi di caricamento dai (due) dischi.

Gli effetti sonori e le musiche sono buone.

IL VOTO

Un'idea originale, variegata ma tecnicamente non perfetta. 7 meno.

Finalmente un gioco in cui non bisogna ammazzare nessuno per giungere alla fine

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade
Softhouse: The Edge's

SNOOPY

Snoopy, il bracchetto disegnato da *Charles M. Schultz*, è certamente uno dei personaggi più noti nel mondo dei fumetti insieme agli altri coprotagonisti *(Peanuts)*: Linus, Charlie Brown, Lucy, Schroeder.

IL GIOCO

Linus ha smarrito la sua vitale coperta e *Snoopy* il bracchetto si incarica di recuperarla.

Questa semplice idea è la base per ambientare in una grafica veramente da cartone animato (vedi foto) una grande serie di piccoli rompicapo da risolvere e situazioni da affrontare per giungere allo scopo finale.

Come in un adventure, si trovano in circolazione parecchi oggetti, che Snoopy può raccogliere con un semplice movimento del joystick verso il basso. Tutti gli oggetti hanno uno scopo, che deve essere scoperto, ovviamente, dal giocatore entro i 45 minuti di tempo concessi per la soluzione. E' importante notare che il gioco ha almeno due possibili soluzioni, in quanto alla partenza la coperta smarrita può essere nascosta casualmente in almento due posti differenti. Si noterà anche durante il gioco che gli altri personaggi (ci sono tutti i Peanuts) si spostano, anche se non sono animati, consentendo di movimentare e variare il gioco.

LA TECNICA

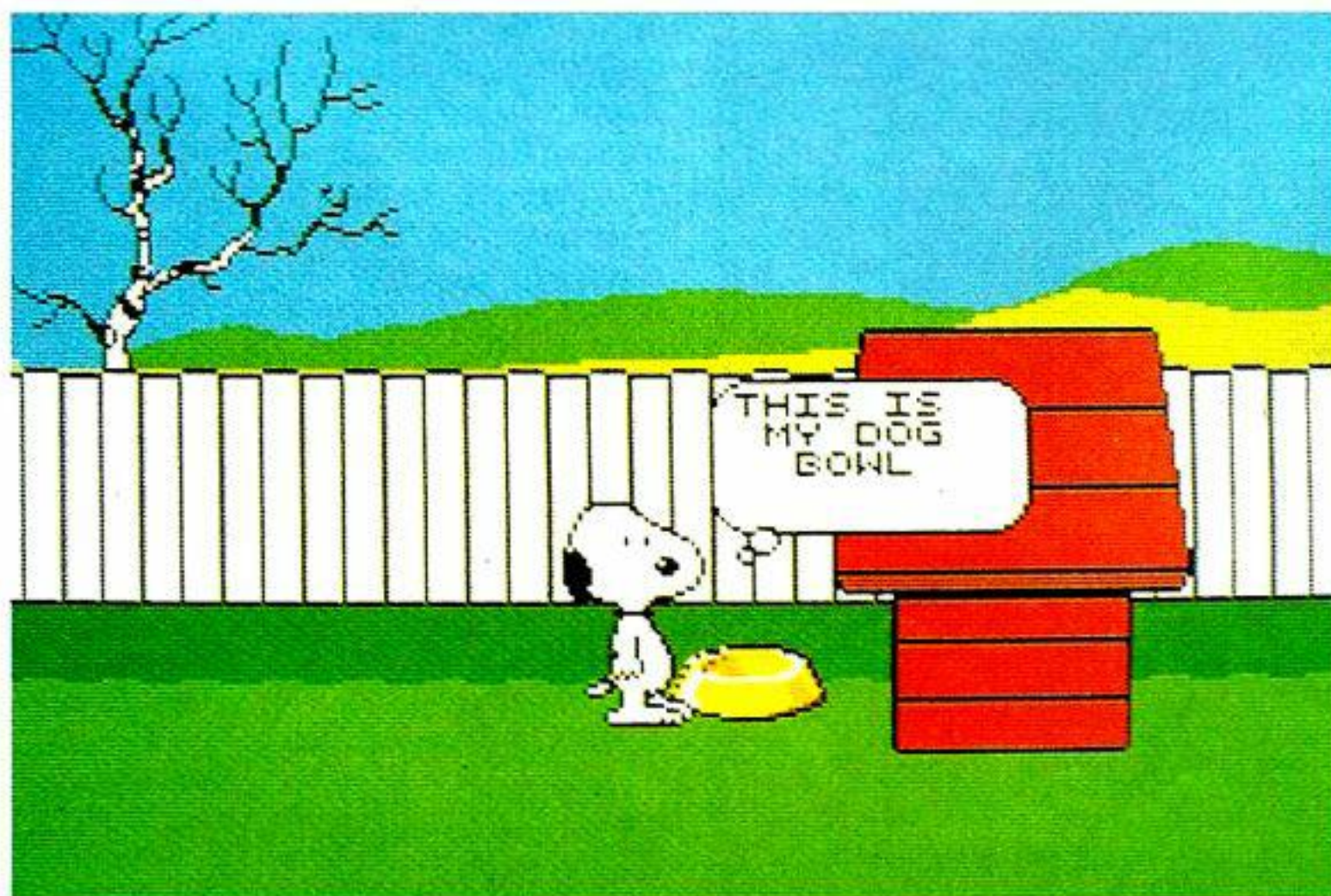
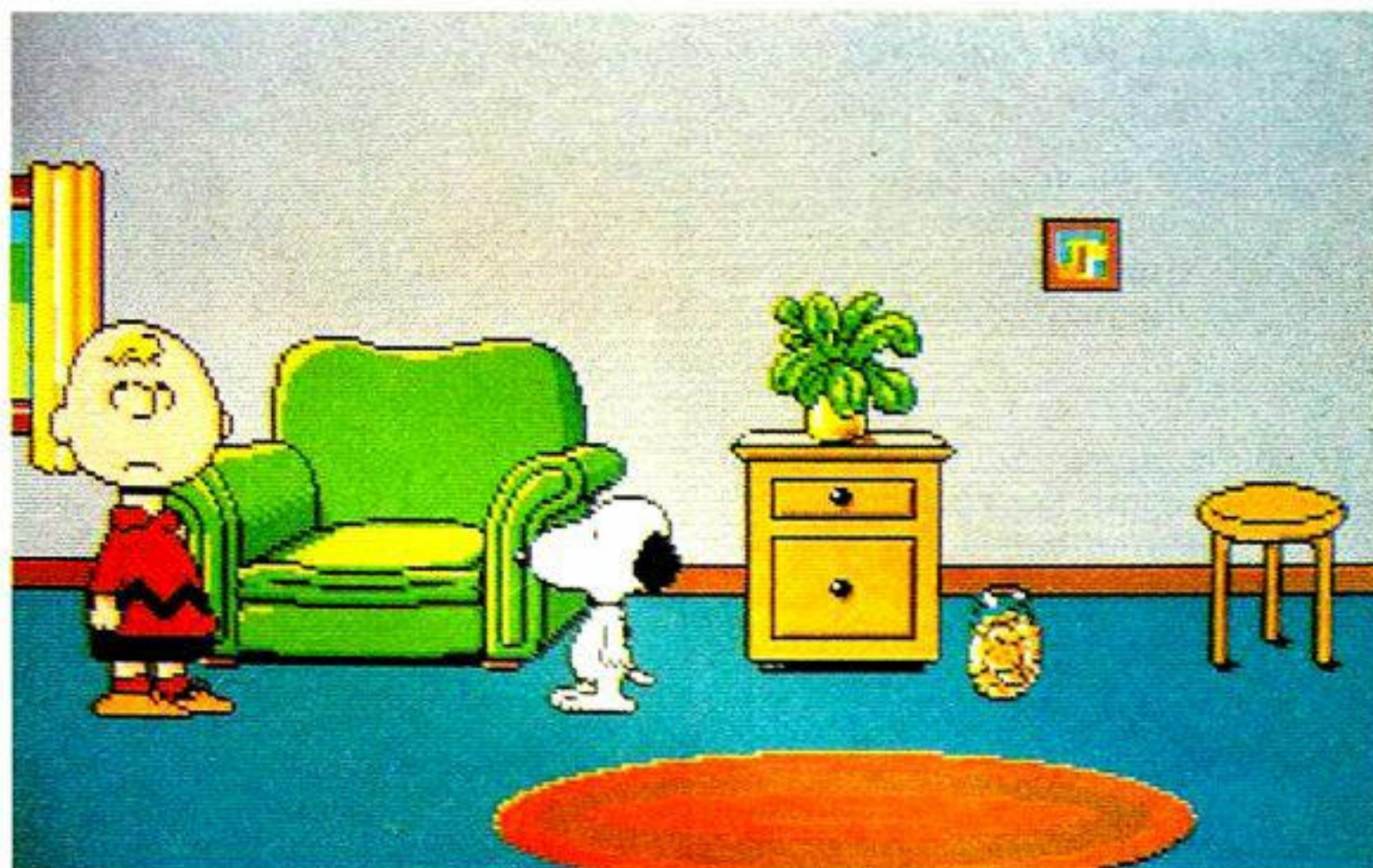
Gli sprites sono grandi e disegnati in maniera ineccepibile, sia per quanto riguarda gli sfondi (belli come i cartoni animati originali) che comprendono anche gli altri Peanuts immobili, sia per quanto riguarda le animazioni di Snoopy, nuvolette, ranocchi, gatti...

E' da notare che non vi sono normalmente tempi di attesa nel passaggio da una schermata alla successiva.

Gli effetti sonori sono ridotti, ma ben fatti. Il controllo del bracchetto col joystick è quanto di più semplice ed immediato si potesse pensare.

IL VOTO

Un gioco che ripone la maggior parte del proprio fascino nella grafica di ottimo livello e nella simpatia dei celeberrimi protagonisti, pur essendo molto semplice nella trama. 7 e mezzo.

WEST PHASER

Nei panni di un cacciatore di taglie mostriamo la nostra bravura nel Far West

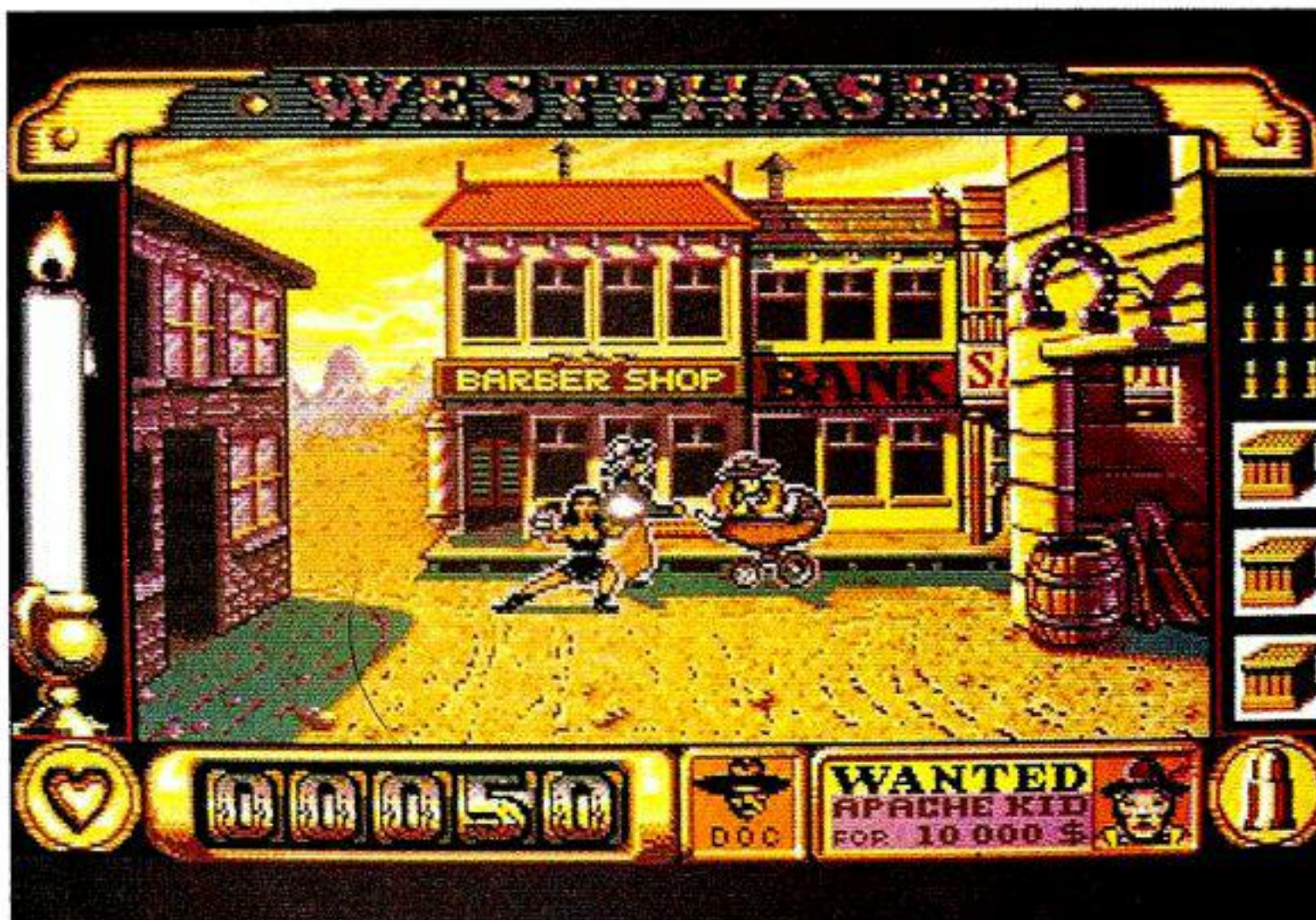
Computer: Amiga inespanso
Gestione: Mouse
Tipo: Arcade
Softhouse: Loriciels

Un classico *spara a tutti con il revolver* ambientato nel Far West distribuito da una vivace ed intraprendente softhouse francese è quello che ci vuole per calarsi nei panni di un *bounty killer*.

IL GIOCO

Inizialmente si sceglie il ruolo che si vuole ricoprire indicando il viso di uno tra otto giustizieri.

Poi si passa alla scelta dell'ambientazione *(California, Nevada, Utah, Colorado, Nuovo Messico od Arizona)* e del lavativo di turno tra sei differenti brutti ceffi, dei quali il più quotato e difficile da stendere è il solito *Billy The Kid*.

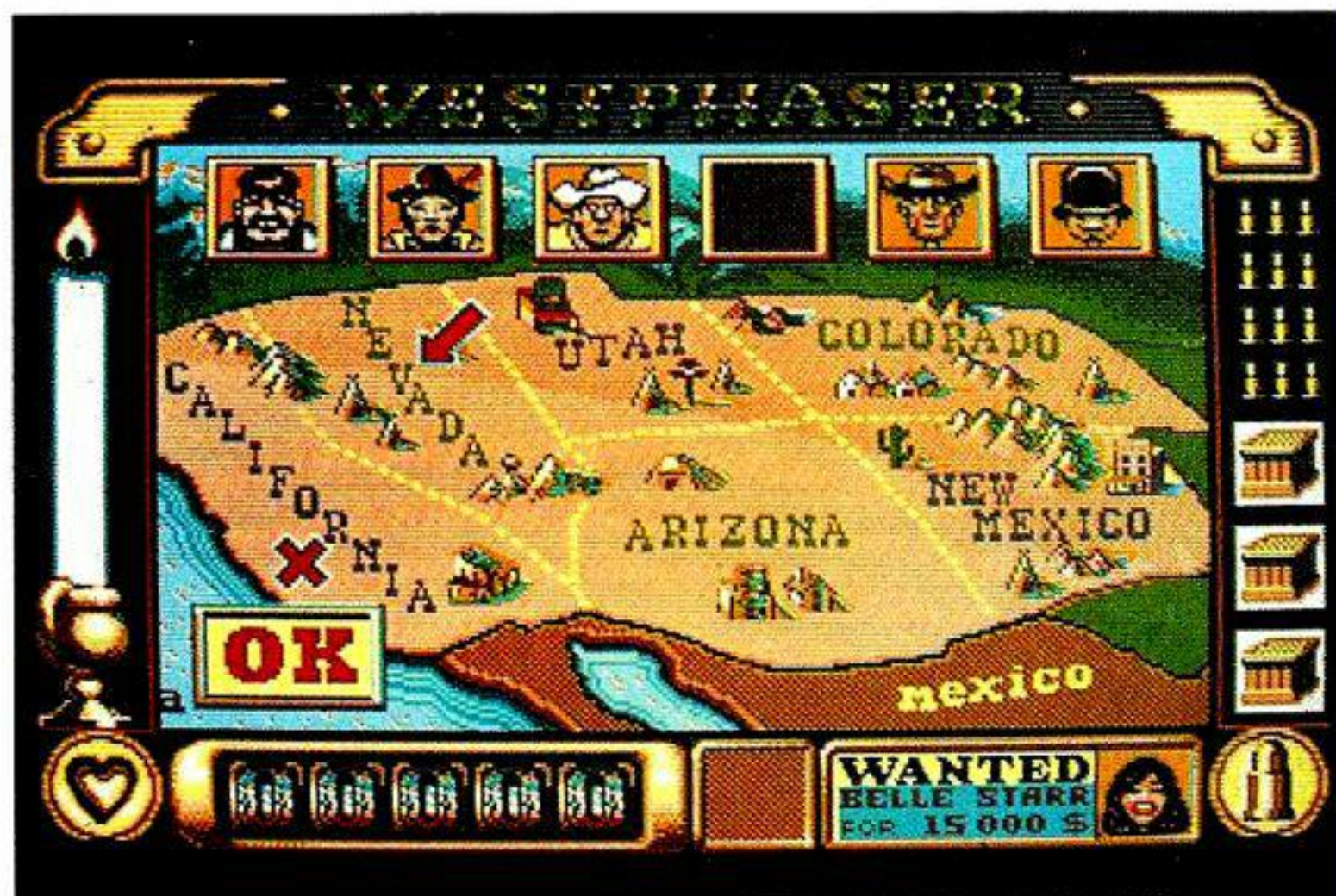
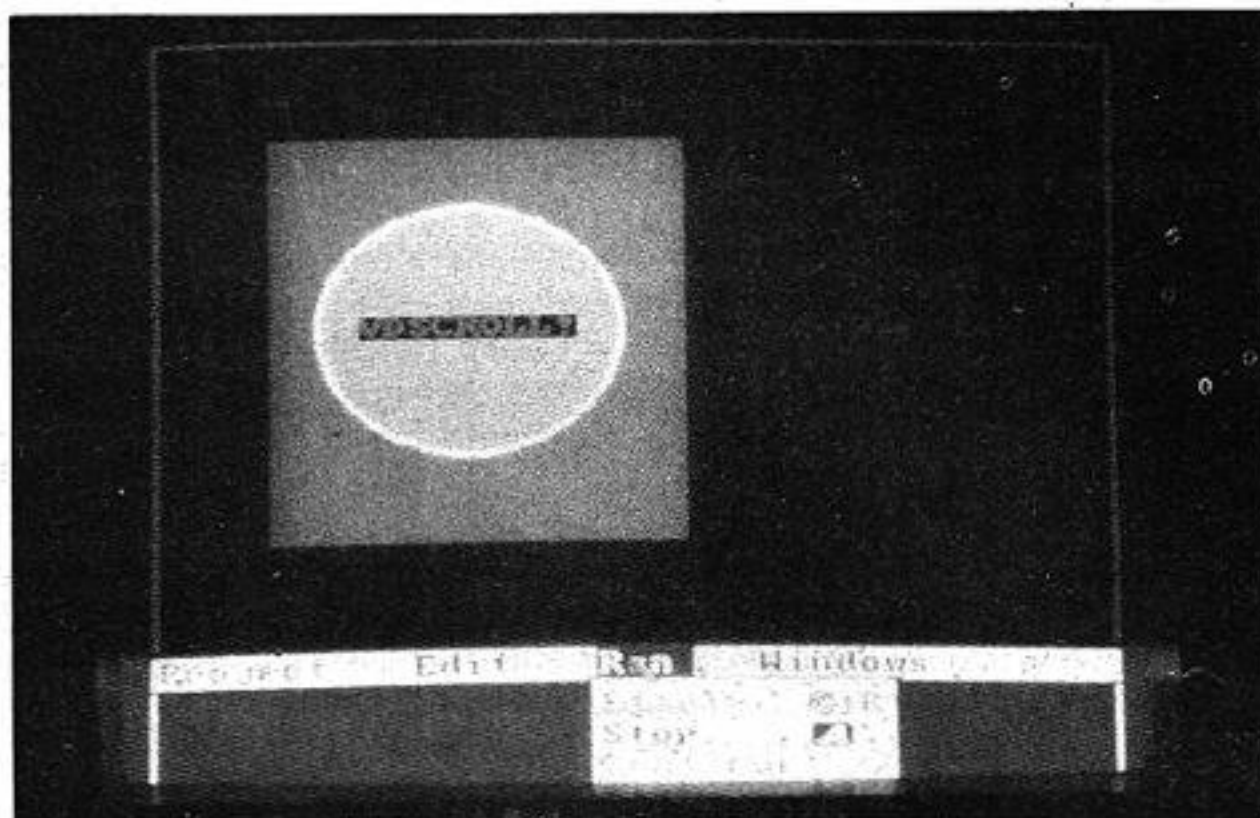
Le scelte si fanno sempre "sparando" col mouse sul gadget o sulla figura interessata. Dopo le scelte si entra nel gioco vero e proprio. Inizialmente l'ambientazione è in una tipica strada del Far West, con case costruite con travi di legno, saloon a porta sbattente e varia fauna umana più o meno armata. In pratica bisogna sparare a tutto ciò che si muove, o minaccia di muoversi, tranne qualche povero innocente chiaramente (quasi sempre) riconoscibile, che se ucciso ci fa perdere dei punti.

Si deve però sparare con discernimento, in quanto le munizioni non sono illimitate.

LA TECNICA

La grafica è curatissima in tutti i punti, non per niente il gioco, concettualmente piuttosto semplice, è distribuito su due dischi (da scambiare solo all'inizio di ogni partita). Le scelte vengono eseguite col mouse su cartine rifinite, con gadget rappresentanti facce ed un solo effetto sonoro (lo sparo).

Il pezzo forte è costituito dalle divertenti animazioni del gioco vero e proprio, in cui parecchi personaggi, molto pittoreschi, sparano da tutte le parti, si rotolano per terra, fanno girare le colt coi pollici prima di sparare, spuntano dappertutto velocissimamente.

IL VOTO

Un simpatico gioco, tecnicamente valido, anche se certamente non originalissimo. 7 meno.

SCHERMO SU SCHERMO

Un facile scroll, tutto in Basic, per migliorare la presentazione grafica di qualunque programma

di **Gregor Samsa**

Intuition, la libreria che gestisce l'interfaccia utente più utilizzata, si presta perfettamente ad un comodo apprendistato.

Non è che sia meno complessa di altre, ma alcune sue funzioni sono accessibili da Basic con molta facilità, per di più consentendo prestazioni difficilmente implementabili con i soli comandi dell'interprete della Microsoft (AmigaBasic, insomma).

Richiamare una Libreria da Basic, non è però altrettanto automatico: si rende indispensabile qualche operazione preliminare.

In pratica, dato che ci accingiamo ad utilizzare Intuition, sarà necessario predisporre un file **Intuition.bmap** per il nostro programmino, da collocare (preferibilmente) nella stessa directory ove esso risiede(rà), o nella directory **Libs** del disco utilizzato per il boot (più complicato).

In effetti, si può anche inserirlo altrove (p. es. in **Ram Disk**), purchè se ne specifichi il percorso al momento in cui viene aperta la libreria (ne riparleremo tra breve).

Come più volte ricordato, per ottenere questo tipo di file va utilizzato il programma **ConvertFD**, presente nella directory **Basicdemos** del dischetto **Extras**, seguendo le modalità illustrate più volte su questa rivista.

Soddisfatti i bisogni (non troppo reconditi) del nostro AmigaBasic, non resta che lanciare il Basic, copiare il listato di queste pagine e salvarlo debitamente su disco: qualche Guru (di troppo) potrebbe venire a curiosare nelle nostre attività.

Si badi che la prima istruzione del programma...

Library "intuition.library"

...presuppone che il file Intuition.bmap sia presente, come già detto, nella stessa directory o in Libs del disco di lancio: se così non fosse, si completi il nome tra virgolette con il suo percorso completo.

Se, ad esempio, il file Intuition.bmap si trovasse nella directory **MieLib** del disco **Work**, il comando andrebbe così modificato:

Library "Work: MieLib/ intuition. library"

Il breve programma, come si può constatare mandandolo in esecuzione, è più che altro un semplice Demo grafico, ma svolge il suo compito grazie ad un sottoprogramma (o, se preferite, **funzione**) completamente "autonomo", estraibile dal suo contesto, e quindi facilmente sfruttabile in applicazioni più personali.

Ma procediamo con ordine.

Dopo aver selezionato Start dal menu Run, viene subito richiesta in input una non meglio precisata velocità.

Si digiti un numero non troppo alto (p. es. 3), e Return.

Lo schermo scrollerà dolcemente verso il basso, scoprendo una seconda videata posta "dietro".

Alla pressione di un tasto, il sipario tornerà in alto con la stessa progressione, in attesa di un'ulteriore pressione di un tasto, che farà tornare in ambiente Basic.

Si eviti, durante l'esecuzione del programma, di premere il pulsante sinistro del mouse, che potrebbe rendere inefficace la pressione dei tasti; qualora si verificasse tale eventualità, basterebbe comunque ricorrere ai menu del Basic per uscire dal programma, oppure tirare giù gli schermi col mouse fino a scoprire la finestra di output del Basic, clickarvi sopra, e quindi premere un tasto.

UN NUOVO COMANDO

L'effetto "sipario" è ottenuto mediante una nuova funzione, arbitrariamente chamata **VdScroll**, chiaramente individuabile in fondo al listato (Sub.... End Sub).

In pratica, svolgerà il suo compito ogni qualvolta viene richiamata con la sintassi...

Vdscroll velocita%, indirizzoschermo&

La variabile **Velocita%** rappresenta un valore intero "corto", cioè composto di due byte, e può anche essere negativo.

In quest'ultimo caso, lo scroll avverrà dal basso verso l'alto.

Chiaro che, per funzionare con valori negativi, lo schermo da scrollare dovrà già essere abbassato (anche di un po'), altrimenti non produrrà alcun effetto.

La seconda variabile, **Indirizzoschermo&** (riferita allo schermo da scrollare) richiede qualche approfondimento.

Giocando con Intuition, è possibile effettuare un gran numero di operazioni, come, tra l'altro, aprire e chiudere nuovi schermi e finestre, spostarli, o disegnarvi sopra.

Alla base di tutto c'è sempre una "Struttura", ovvero un gruppo di dati (numerici) che risiedono da qualche parte in memoria, e che la Libreria adopera per riconoscere schermi, finestre, ecc.

Queste strutture possono essere create da programma, per renderle assolutamente personali, ma... non è il caso di entrare nei particolari, o ne verrebbe fuori un bel trattato (lo si farà in altra sede, se abbondantemente richiesto).

Al momento, tutto ciò che interessa sapere, è che ad ogni schermo, anche se creato tramite l'istruzione Screen del Basic, corrisponde una di queste strutture.

Ed è proprio l'indirizzo in memoria di questo gruppo di dati che dobbiamo fornire alla nostra funzione (vedremo tra poco perchè).

Per ottenerlo, è necessario aprire una finestra (istruzione **Window**) sullo schermo in questione.

Anche la finestra, come già detto, possiede una sua Struttura in memoria.

L'indirizzo di quest'ultima, è però facilmente ottenibile, dato che esiste (per fortuna!) la funzione Basic **Window(7)** (vedi manuale).

Ebbene (Macchiavelli, al confronto, era un principiante), al byte 46 di questa struttura è memorizzato, in formato Intero Lungo (= 4 byte), l'indirizzo di inizio della struttura-schermo.

In definitiva, la variabile che interessa può essere perciò ricavata così...

Indirizzoschermo& = PeekL (window (7) + 46)

...come riscontrabile nel listato (variabile **sch&**).

VARIAZIONI SUL TEMA

Riassumendo, affinchè il nuovo comando VdScroll funzioni correttamente, il programma che ne fa uso deve quindi necessariamente provvedere a:

1) Aprire la libreria Intuition.library, ricordando (eventualmente) di specificare il percorso per rintracciare il file Intuition.bmap precedentemente realizzato.

2) Aprire uno schermo (quello da scrollare), e su di esso una finestra (che poi, volendo, potrà essere richiusa), quindi adoperare la formuletta appena vista per rintracciarne l'indirizzo di struttura.

3) Invocare la funzione secondo la sua corretta sintassi.

Alla base del nuovo comando, come rilevabile nella Sub del listato, è la routine **Movescreen** della Intuition Library.

Questa viene richiamata adoperando il comando **Call**, in quanto non restituisce alcun valore; per lo stesso motivo, al momento di aprire la libreria, non è necessaria una preventiva istruzione **Declare Function**.

Movescreen funziona solo per scroll verticali, ma è necessario ugualmente inservi un parametro riguardante il movimento orizzontale (0 nel listato).

La sua completa sintassi di chiamata è:

Call Movescreen (indirizzoschermo&, X%, Y%)

...con X ed Y che indicano di quanti pixel per volta deve essere spostato lo schermo in senso orizzontale (non implementato) oppure verticale.

Nel programma demo, viene banalmente disegnato uno schermo "sottostante".

A parte l'ovvia possibilità di migliorarne la grafica, magari caricando una immagine di Deluxe Paint (se ne riparlerà...), lo scroll potrebbe essere effettuato sullo schermo disegnato: basterebbe eliminare la riga contenente l'istruzione Screen 3...

Analogamente, si potrebbe rendere "variopinto" lo schermo che fa da sipario, o ancora fare scrollare in successione più schermi: insomma, c'è proprio di che sbizzarrirsi.

Sulla base di quanto visto, diventa inoltre molto semplice aggiungere altre feature ai nostri programmi.

Una volta capito come ottenere l'indirizzo in memoria della struttura di uno schermo, è infatti immediata la possibilità di attivare altre due funzioni di Intuition che non necessitano di altri parametri.

Si sta parlando di **Screentoback (indir&)** e **Screentofront (indir&)**, che, rispettivamente, portano indietro ed avanti lo schermo il cui indirizzo è specificato come parametro.

Per non restare sul vago, si provi ad aggiungere nel listato, subito sotto la seconda chiamata a Vdscroll (e prima della label Fine:), queste due righe:

GOSUB attesa
CALL screentoback(sch&)

Lanciato ancora una volta il programma, si vedrà come, dopo il risollevarsi dello schermo-sipario, l'ulteriore pressione di un tasto provocherà l'immediata ricomparsa dello schermo sottostante.

A parte le possibilità pratiche del comando Vdscroll, non si sottovaluti l'importanza di quanto acquisito, anche se le capacità di Intuition non si fermano certo qui.

Applicazioni più sofisticate fanno infatti pendere l'ago della bilancia in favore del C o dell'Assembly.

Ma anche adoperando tali linguaggi, e nonostante le ovvie differenze, l'uso di Librerie e Strutture rimane pressochè identico.

Per chi non mira così in alto, c'è comunque un Basico Vdscroll con cui consolarsi.

BASIC; TROPPO FORTE

Secondo un luogo comune di (quasi) universale accettazione, il Basic è figlio bastardo di più nobili e titolati linguaggi.

L'affermazione, tuttavia, è solo apparentemente limitativa.

Da un punto di vista "fisico" contiene certamente un minimo di verità: nessuno può negare che lo sviluppo (abnorme?) di tale linguaggio era in origine previsto in funzione di una utenza dilettantistica.

Di contro si è visto come, a seconda delle macchine sulle quali è implementato (ed in ossequio ad una corretta logica di evoluzione) proprio il Basic è andato assumendo connotazioni simili a quelle di altri linguaggi.

Senza, ovviamente, pensare ad impossibili accostamenti con il linguaggio macchina (mai mischiare il sacro con il profano), basti ricordare l'assunzione, p. es. nel Commodore 128, di possibilità tipicamente "Pascaliane".

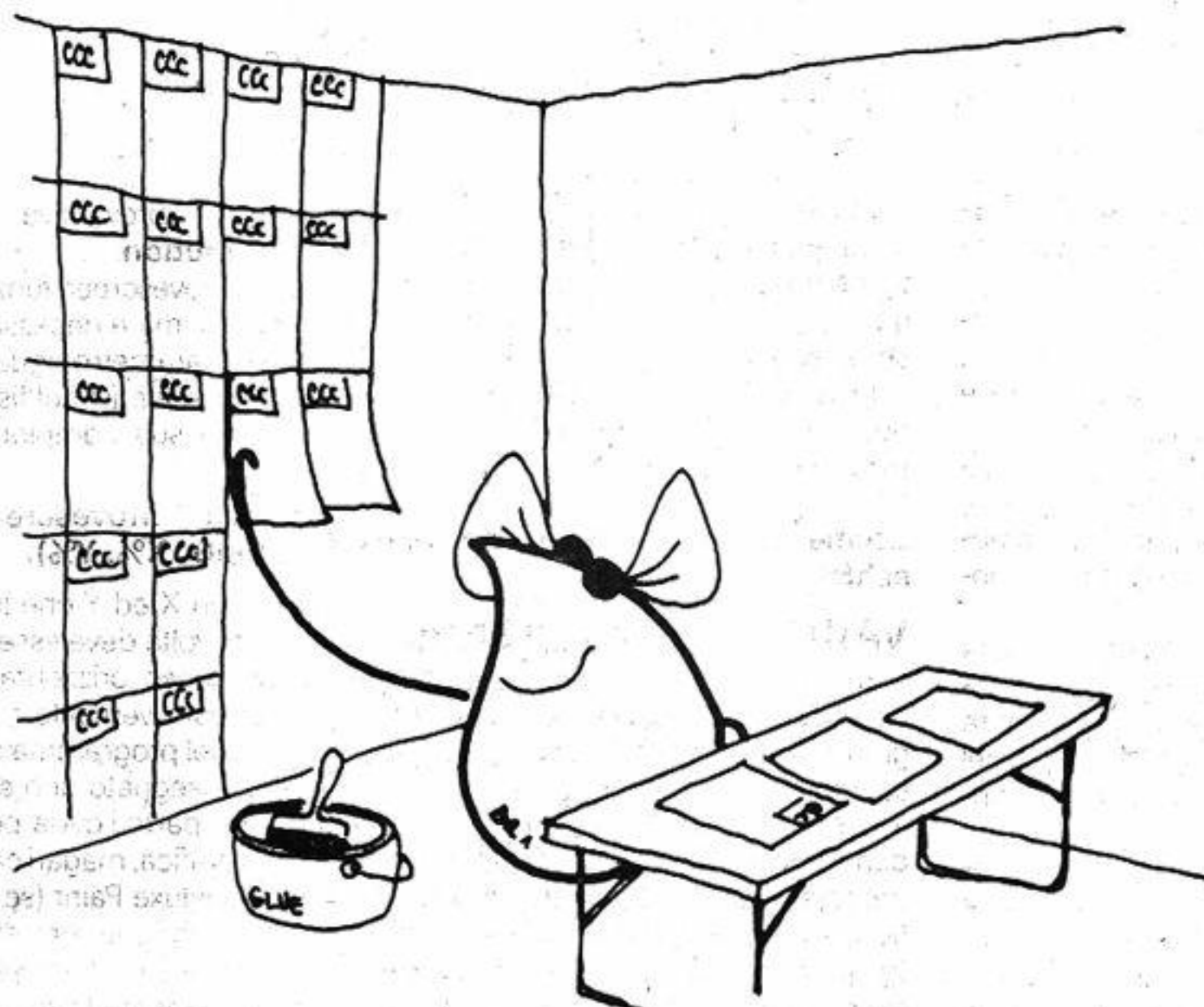
Con Amiga, anzi con **AmigaBasic**, lo stesso "Transfert" si è sviluppato nei confronti del **C**, linguaggio di fatto molto più vicino alla struttura del computer... ma anche più complesso.

Chi non ha voglia (o tempo) di scontrarsi con l'esuberante punteggiatura di quest'ultimo linguaggio, non ha però di che lamentarsi.

A parte l'inevitabile, e spesso ingombrante, presenza dell'interprete, AmigaBasic può anch'esso sfruttare molte (non tutte!) delle risorse nascoste del nostro amato 16 bit, a patto di sapersi aggirare (un po') tra le cosiddette Librerie di sistema.

Difficile?

L'importante è cominciare.

Naturalmente dalle cose più semplici, anche se non necessariamente meno utili.

```
'=======================================
'    DEMO FUNZIONE BASIC "VDSCROLL"
'=======================================
'
'sintassi 1:
'CALL Vdscroll (velocita%,indir.schermo&)
'sintassi 2:
'Vdscroll velocita%,indir.schermo&
'
'=======================================

'--------- apertura libreria -----------
LIBRARY "intuition.library"
'---------------------------------------

SCREEN 2,320,250,2,1:WINDOW 2,,,0,2

'---- indirizzo schermo di scroll -----
SCREEN 3,320,250,1,1:WINDOW 3,,,0,3
sch&=PEEKL(WINDOW(7)+46)
IF sch&=0 THEN
  PRINT "ERRORE! PREMI UN TASTO"
  GOTO fine
END IF
'---------------------------------------

INPUT"velocita'";v%:CLS
WINDOW OUTPUT 2
PALETTE 0,0,0,0:PALETTE 1,.93,.2,0
PALETTE 2,.33,.87,0:PALETTE 3,1,.13,.93
LINE (35,35)-(170,170),1,bf
CIRCLE(100,100),50,3:PAINT(100,100),2,3
LOCATE 13,9:PRINT "VDSCROLL!"
LOCATE 27,20:PRINT "PREMI UN TASTO"

'---------------------------------------
Vdscroll v%,sch&
'---------------------------------------

GOSUB attesa
WINDOW OUTPUT 3
LOCATE 27,20:PRINT "RIPREMI UN TASTO"

'---------------------------------------
Vdscroll -v%,sch&
'---------------------------------------

fine:
GOSUB attesa

WINDOW CLOSE 3:WINDOW CLOSE 2
SCREEN CLOSE 2:SCREEN CLOSE 3
LIBRARY CLOSE:END

attesa:
x$=INKEY$:IF x$=""THEN GOTO attesa
RETURN

'======== FUNZIONE VDSCROLL ===========

SUB Vdscroll (velocita%,schermo&) STATIC
y%=260/velocita%
IF velocita%<0 THEN GOTO su



 FOR x%=0 TO y%
  CALL movescreen(schermo&,0,velocita%)
 NEXT
 GOTO torna
 su:
 FOR x%=y% TO 0
  CALL movescreen(schermo&,0,velocita%)
 NEXT
 torna:
 END SUB

'=======================================
```

POSTAMIGA

(A cura di Domenico Pavone)

ARDUA FORMATTAZIONE

Sono un novello utente di Amiga 500, e mi sono scontrato con una difficoltà che sembra insormontabile: Non riesco a formattare i dischetti nuovi in alcun modo, neanche da Workbench. Faccio presente che non posseggo il secondo drive e che, provando ad usare l'opzione Initialize del menu di Workbench, ottengo solo segnalazioni tipo Disk Corrupt, Df0: Bad, oppure Error While Opening Sys: System / Format. Qualche volta tutto sembra partire correttamente, appare la finestra con la scritta Formatting Cylinder ...79, il 79 diventa 78 e poi il procedimento si blocca. Ho provato ad usare DiskDoctor, ma senza risolvere il problema. Il colmo è che invece la copia di un disco su uno nuovo riesce perfettamente! Che cosa posso fare?

(Matteo Diterlizzi - Milano)

Anzitutto, DiskDoctor non c'entra proprio niente in quanto serve a ricostruire il contenuto di un dischetto danneggiato, ma sempre che esso sia già nel formato *AmigaDos*, ovvero preventivamente formattato e (più o meno) infarcito di dati, files, eccetera.

Il fatto che la copia riesca, inoltre, fa supporre che non ci siano problemi di malfunzionamenti hardware.

Prova dunque a seguire scrupolosamente questa procedura (per un solo drive), badando che il disco Workbench sia una copia integrale dell'originale, senza alcun tentativo di "personalizzazione".

1) Dopo aver caricato il Workbench, estrarre il disco di sistema dal drive ed inserire quello da formattare. Se questo è ancora vergine, la sua icona sarà per forza di cose segnalata dal nome *Df0: bad* (che sta per cattivo, no buono, schifido).

2) Clickare (una volta sola, o appariranno i vari requester *disk corrupt*..., e simili) sulla sua icona, dopodichè selezionare *Initialize* dal menu *Disk* del Workbench. Apparirà il requester: *Please replace volume Workbench1.3 in any drive.*

3) Obbedire, reinserendo il disco di sistema.

4) Dopo qualche ronzio del drive, al nuovo avviso di Amiga *(Please insert disk to be initialized in drive df0:)*, togliere il disco Workbench (ma soltanto *dopo* che la spia si è spenta) e schiaffarvi quello da formattare.

5) All'ennesimo messaggio *Ok to initialize...*, clickare su *Continue*, ed attendere che il lavoro venga ultimato.

E così sia.

Se le varie fasi sono eseguite alla lettera, e con un disco di sistema *non* manipolato, tutto dovrebbe andare per il verso giusto. Per precauzione, si controlli che il disco Workbench sia protetto dalla scrittura (tacca aperta).

Un pò perplessi lascia invece l'eventualità che il procedimento di formattazione si avvii regolarmente, e poi non continui dopo l'iniziale segnalazione *Formatting Cylinder.*

In questo caso, occorre controllare che non sia opera di un *virus* (tramite un programma virus killer), soprattutto se il dischetto da formattare era già usato oppure si era adoperato in precedenza il floppy Workbench non protetto in scrittura, magari dopo aver giocato un po' con qualche game di provenienza dubbia.

Per la cronaca, il vecchio ma sempre diffuso Byte Bandit Virus, provoca (tra le altre cose) proprio questo tipo di anomalia.

DOVE SAVE?

Non riesco a registrare programmi basic su un disco nuovo. Al momento del save, il computer mi chiede il disco Extras per salvarli lì!

(Carmine Esposito - Napoli)

Per superare la difficoltà, anche in questo caso legata soprattutto alla mancanza di un secondo drive (ma la vogliamo capire che è quasi indispensabile per lavorare più tranquilli e distesi?) basta far precedere, al nome del programma da salvare, quello del disco voluto.

Se, per esempio, si intende archiviare il nostro bel file basic in un disco cui abbiamo assegnato il nome *MieiLavori*, basterà immettere nella finestra del *Save*...

MieiLavori:nomeprogramma

...badando a non dimenticare il simbolo due punti(:) e senza lasciare alcuno spazio prima e dopo di esso.

Per ricaricare il file nell'editor occorrerà, come ovvio, fornire lo stesso tipo di indicazione (definita *Path*, cioè *Percorso* del file).

Come alternativa, dopo essere "entrati" in ambiente Basic, si può impartire (direttamente nella finestra di output) il comando:

Chdir "MieiLavori:"

Dal questo momento in poi qualunque operazione di lettura o scrittura sul drive, verrà *forzata* verso il disco (o directory) desiderato.

E' chiaro che, in questo caso, non sarà più necessario specificare nel nome dei file l'intero percorso.

COMINCIARE CON IL C

Ho deciso di imparare il linguaggio C, così mi sono procurato un compilatore ed ho iniziato a leggere il testo di Kernighan & Ritchie, provando poi sul computer i risultati dei miei esperimenti. Sono riuscito ad ottenere qualche risultato, ma il mio lavoro si è arenato nell'impossibilità di capire e conoscere le decine di Flag e strutture che si trovano nei file di inclusione...

(Paolo Flora - Paluzza)

La lettura del *Kernighan & Ritchie* (per i meno esperti: i padri fondatori del linguaggio) è senz'altro un passo (quasi) obbligato per conoscere il linguaggio **C**, ma è altrettanto importante accostare il linguaggio alla struttura della macchina, nel nostro caso l'Amiga.

Il che, tradotto in parole povere, significa avere la possibilità di documentarsi con esattezza su librerie, strutture, device, hardware, eccetera.

A tale scopo, ed il discorso non si limita al solo C, è pressoché indispensabile acquistare i quattro (o tre nell' ultima revisione) volumi *Reference Manual* della *Addison - Wesley* (in inglese), ognuno dei quali descrive: *Libraries and Devices, Exec, Intuition, Hardware.*

A questi, inoltre, è consigliabile aggiungere un qualche testo didattico non finalizzato all'apprendimento generale del C, ma specificamente dedicato all'Amiga.

Ogni buona libreria, e spesso anche i negozi di articoli informatici, propongono in genere una discreta scelta di titoli al riguardo, anche in italiano.

Il tutto, come intuibile, comporta un non indifferente danno... alla tasca.

Per chi non potesse (o volesse) sottoporsi a questo volontario salasso, non resta che affidarsi a quanto rintracciabile tra le pagine delle varie riviste del settore.

SGANCIARSI DAL WORKBENCH

Ho copiato alcuni files in un floppy che vorrei adoperare come "disco lavoro", ma non riesco a farli partire. Esiste un modo per farlo, senza dover necessariamente ricorrere al Workbench? E' vero che esiste un sistema chiamato Startup-Sequence?

(Claudio Canis - Savona)

Se il problema è limitato alla possibilità di lanciare un file presente in un altro dischetto, valgono le stesse considerazioni espresse nella risposta precedente, anche se non si tratta di files basic.

Se, invece, si vuole rendere indipendente il disco che li contiene, il discorso si complica un po', e non può essere esaurito in questa sede.

Grosso modo, comunque, si devono effettuare le seguenti operazioni:

Copiare le directory *L, Libs* e *Devs* prelevandole dal disco Workbench.

Creare una directory *S* (comando *Makedir Nomedisco:S)* ed editare un file (non un "sistema") di nome *Startup-Sequence* che contenga la procedura di lancio; diciamo che, ridotto proprio al minimo, basterà che contenga le sole due istruzioni...

Loadwb
Endcli

Creare una directory *C*, e copiarvi almeno i comandi adoperati nella startup-sequence, oltre ad *Execute* e *Run* (il tutto va prelevato dalla directory *C* del disco di sistema).

Installare il dischetto, ovvero digitare da *Shell* il comando *Install Nomedisco:*

Per chi è alle prime armi, non sarà proprio come bere un bicchier d'acqua: nella schematica descrizione, oltre ad alcune difficoltà intrinseche, si nascondono infatti numerosi trabocchetti, superabili solo con un minimo di esperienza.

Un consiglio: cimentarsi nell'impresa *solo* dopo aver approfondito le cognizioni che riguardano l'uso di *Shell* e del *Dos* in generale.

Per chi si sente già pronto, o vuole comunque provarci, si ricorda che l'argomento è stato trattato in modo molto più approfondito sul non lontano numero 68 (ottobre '89) della rivista, articolo *Workbench lavori in corso* (eventualmente disponibile presso il nostro servizio arretrati).

DE GURIS

Ogni tanto, usando il mio Amiga 500 inespanso, appare il fatidico "Guru Meditation".
A parte gli altri vantaggi derivanti dall'avere un mega di memoria, il problema si risolverebbe se acquistassi l'espansione?

(Antonio De Giorgi - Marciano di Leuca)

Il Guru, caro lettore, è una bestia fedele: non abbandona mai il suo amato padrone.

Con l'espansione vengono di certo ridotti i *tilt* legati alla scarsa disponibilità di memoria, ma, ahinoi, non sono solo questi i motivi che portano al blocco di Amiga.

Rimedi?

Una buona dose di rassegnazione, sperando in tempi migliori (Kickstart 1.4?), ma nel frattempo fornendosi di qualche programma tipo *Gomf* oppure *Noguru*, che consentono almeno di limitare i danni.

PRINTER OK

Posseggo una stampante Commodore Mps 1250, che adopero con il mio C/64. Visto che penso di passare ad Amiga, sapreste dirmi se esistono problemi di compatibilità?
(Marco Alzetta - Pordenone)

Nessun problema, il connubio è reso possibile dall'interfaccia *Centronics* di cui è dotata la stampante.

Inoltre, a maggior garanzia, nei dischetti di sistema esiste un driver specifico proprio per la *MPS 1250*, in emulazione *EpsonX*.

P.S. La lettera, come del resto avviene per tutte (anche quelle non pubblicate), è stata "letta con attenzione".

ORIGINALI?

Con alcuni giochi originali che possiedo, mi capita di veder apparire un request "Please insert volume L" e lo start si blocca; oppure "Error validating disk" e il consiglio di ricorrere a Diskdoctor: dopo averlo fatto, però, non si è riparato nulla.
(Liborio Di Micell - Corleone)

Il secondo problema, può essere dovuto ad involontari maltrattamenti del floppy, oppure a *precoci* estrazioni dal drive mentre questo è ancora in funzione.

In questi casi, *Diskdoctor* funziona solo se il disco non è dotato delle protezioni che

ASSEMBLY CONTRO C

Sono un "ex" del C/64, e credo di aver fatto uno sbaglio ad imparare il linguaggio macchina per quel computer (seguendo i vostri consigli), visto che su Amiga sono costretto a studiare tutto daccapo. Meglio dedicarsi al basic, o al C, che almeno è standardizzato su tutti i computer.
(Giuseppe Palermo - ??)

Se, come dici, hai imparato bene l'*Assembly* "ad otto bit", allora non può certamente esserti sfuggita l'introduzione che normalmente viene fatta a tale linguaggio, tanto nei testi specializzati che nelle pagine della nostra rivista.

Dove, ben in evidenza, è stato sempre precisato che il linguaggio macchina è il meno "trasportabile" in assoluto, visto che, di norma, dialoga direttamente con il microprocessore specifico per il tipo di computer. Il che, come ben saprai, comporta vantaggi e svantaggi.

Non è però corretto affermare che quanto in precedenza appreso "non serve a niente": anche se i codici dell'Assembly *68000* saranno diversi da quelli del *6502*, un byte è pur sempre un byte, e la sua manipolazione, con qualche doveroso approfondimento, non è poi così differente.

Con questo non si vuol dire che con Amiga siano tutte rose e fiori, ma ciò che si è imparato su un C/64 non è da sottovalutare.

Se non altro, evita di dover apprendere ex novo le basi teoriche del linguaggio.

Un *Branch* o un Jump to subroutine *(Jsr)*, o uno scorrimento di bit, per esempio, non saranno una novità assoluta, come pure l'uso di *Macro Istruzioni* in un sorgente sviluppato con un buon pacchetto *Assembler*.

Paradossalmente, molti singoli codici dell'Amiga risulteranno un sollievo per gli "ex" del C/64, che per raggiungere lo stesso scopo dovevano creare intere routines

Di indubbia maggiore difficoltà è il cambiamento di "logica" cui si è costretti, soprattutto in rapporto al famoso "multitasking" e alla nuova dimensione a 16 bit.

Inoltre, non bisogna dimenticare che l'Assembly può essere utilizzato in due modi: rivolgendosi direttamente all'hardware, o adoperandolo ad un livello leggermente più alto, sfruttando le routines di sistema (le cosiddette librerie).

Il primo è decisamente ostico e non sempre consigliabile, dati i continui mutamenti apportati al sistema dalla Commodore.

Il secondo, invece, è molto più abbordabile.

Quanto al C, la cui conoscenza (anche approssimativa) è diventata pressochè obbligatoria per chi usa Amiga, è certamente più standardizzato dell'Assembly, ma, a meno che non ci si limiti a qualche *Printf*, implica anch'esso un notevole approfondimento dei meccanismi interni del computer. E, si badi, si sta parlando proprio dei meccanismi particolari di un certo tipo di macchina.

Non si pensi, quindi, che sia poi così facile da... digerire, o che, cambiando computer, tutto rimanga assolutamente identico.

Le preferenze, poi, sono un fatto soggettivo, oltre che legate a particolari applicazioni: c'è chi (indovinate come la pensa il sottoscritto...) detesta la sintassi del C, e chi la trova sublime, ma l'ideale sarebbe conoscere entrambi i linguaggi.

Salvo poi scoprire che, tutto sommato, il buon vecchio basic (non necessariamente Amiga Basic, e non necessariamente interpretato) è più che sufficiente per quello che si intende realizzare.

spesso sono presenti nel software commerciale.

Gli altri inconvenienti, a dispetto di quanto affermato, sono per lo più da attribuirsi a copie *pirata* dei games.

A2000 E MEMORIA

E' possibile ovviare all'inconveniente dovuto alla configurazione degli Amiga 2000 con 1 Mega byte di Chip Ram, per cui alcuni giochi non funzionano, mentre su A500 girano benissimo?

(Daniel San - Firenze)

Ho letto su Postamiga del n. 70 (Dicembre 89) la risposta data ad un possessore di Amiga 2000 che chiedeva come appurare il tipo di Ram di cui disponeva. Seguendo quel consiglio, ho provato a dare il comando Avail, con il seguente risultato: Chip 421696... (ecc,); Fast 392528...(ecc.); Total 814224...(ecc.). Inoltre alcuni giochi che posseggo non funzionano: si tratta dello stesso motivo?

(Bruno Celati - Milano)

Effettivamente, l'introduzione del *Mega* di memoria tutta di tipo *Chip*, ha creato qualche problema di compatibilità con del software che non prevedeva una simile configurazione.

In alcuni casi, l'inconveniente può essere risolto semplicemente aggiungendo delle espansioni di *Fast Ram* al 2000 (non certo solo per qualche giochino, dati i costi).

Come testimoniato dalla seconda lettera, però, non tutti i problemi devono essere attribuiti al Mega di Chip Ram.

Il responso di *Avail* citato da Bruno Celati, infatti, dice chiaramente che nel suo caso la configurazione è quella consueta riscontrabile anche nei 500 con espansione di memoria: 1/2 Mb di Chip e 1/2 Mb di Fast Ram.

Delle incompatibilità, talvolta, possono sorgere anche in rapporto alla versione del Kickstart, in questo caso più spesso per colpa dei programmatori che dei progettisti di Amiga. Per non parlare poi, ed è certamente questa la causa più diffusa (anche se inconfessata), degli interventi "esterni" sul software in modo da renderlo copiabile oppure per eliminare qualche copyright....

TUTTO A SUO TEMPO

Perchè non parlate del KickStart 1.4, che pare verrà immesso presto sul mercato?

(Emanuele - Vimercate)

(Omar Buono - Firenze)

Perchè le "anticipazioni" non servono quasi a niente. Quando verrà effettivamente messo in circolazione ufficialmente, se ne valuteranno a fondo le potenzialità, lo si testerà ben bene, ed infine se ne scriverà a iosa, magari colmando le (abituali) lacune del suo manuale.

Solo così non risulterà solo teoria.

SFOGO AMARO

Vi scrivo questa lettera mentre sono ancora sotto l'effetto della frustrazione e dello svilimento più totali. Dopo essere stato per diversi anni un felice ed esperto "sessantaquattrista", da circa un mese sono diventato un "Amigo", ed ho subito cominciato a smanettare sul mio A-500 inespanso, con qualche risultato. Dopo aver perso molto tempo, non sono però riuscito a venire a capo del vostro articolo "Un sistema a prova di Guru" (n.66), che tratta della Rad. L'argomento mi è sembrato molto interessante, e credo di averlo capito, ma non riesco ad installarla in alcun modo. In particolare, dopo che inizia l'esecuzione, il sistema mi segnale che la Ram è "Full", oppure, dopo vari tentativi di modifica, che non può creare la directory Rad:L. Ma perchè?

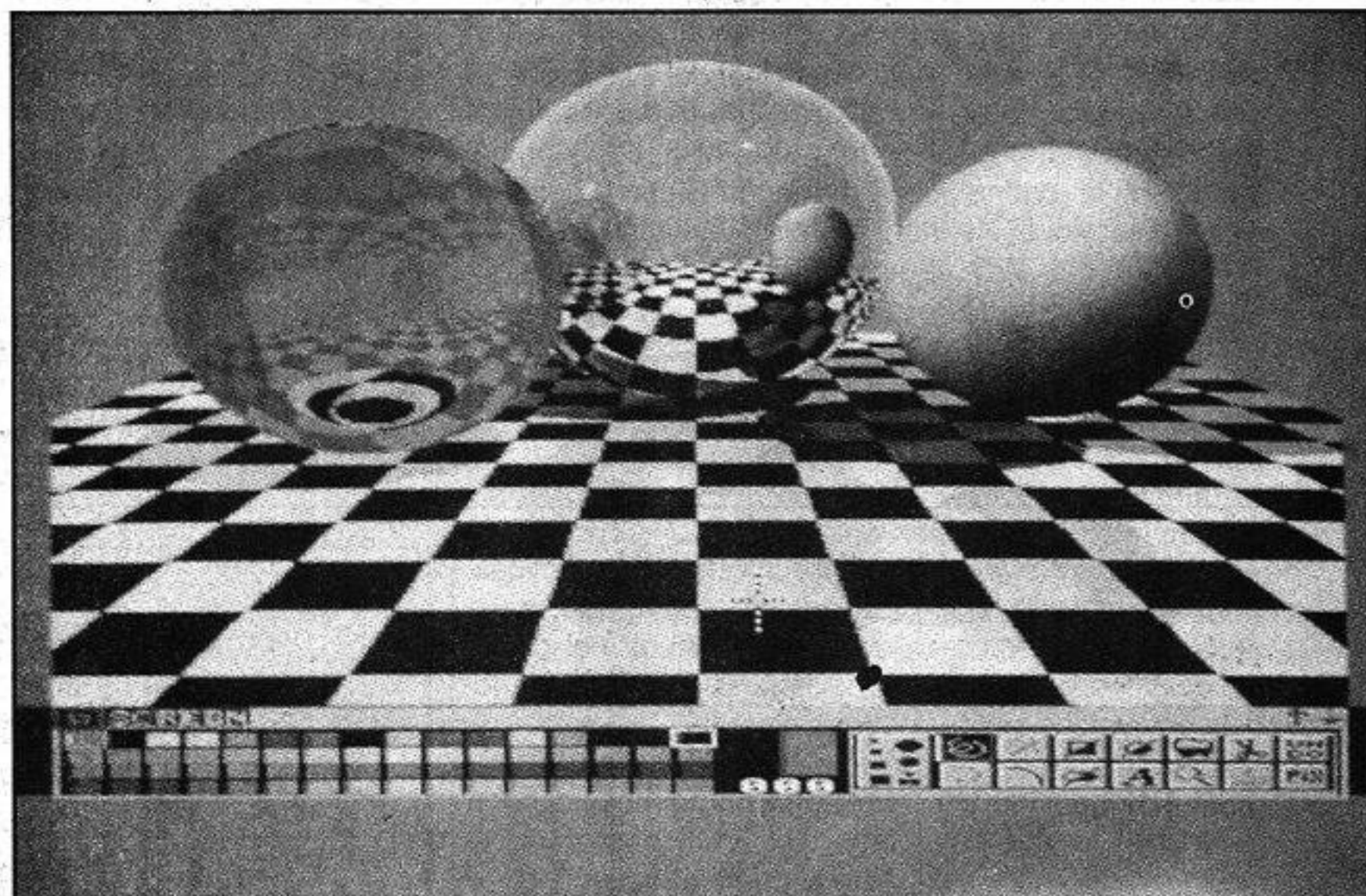
(Tonino Giorgi - Ascoli Piceno)

Coraggio, caro lettore. Non sei il solo ad esserti sentito perso, dopo la decisione di "appendere al chiodo" la familiare tastiera del C/64 o C/128.

D'altra parte, sebbene l'uso generico di Amiga sia estremamente semplice, un approccio più evoluto richiede un minimo di assuefazione alle nuove caratteristiche.

Ma veniamo al problema Rad.

Intanto, con un 500 inespanso, è consigliabile assegnare al parametro *HighCyl* della *MountiList* un valore non troppo alto, non superiore a 29.

Il fatto, però, che nel caso specifico il *file batch* che do-

ACCHIAPPA PARAMETRI

Ho un problema che mi assilla: come posso fare, adoperando il Seka Assembler, a "catturare" i caratteri che andrebbero inseriti a destra di un comando che voglio progettare? E come procedere per aggiungere il byte nullo necessario per passare il parametro alla funzione Write?

(Luigi Marchetto - ??)

Quello accennato è uno dei primi aspetti che di solito si affrontano nella programmazione Assembly, ovvero il prelevamento di uno o più parametri assegnati tramite una linea di comando.

Per essere più chiari: se, in una finestra *Cli* o *Shell* si digita (p. es.) *List df0:*, *List* è un file eseguibile che verrà caricato dalla directory corrente o da *C:*, mentre *df0:* costituisce un parametro che dovrà essere letto dal programma di nome *List*.

La procedura per leggere la stringa posta dopo il nome del file, è più semplice di quanto si possa credere.

Dopo che si preme il Return (o Enter), infatti, Amiga colloca in Ram la stringa-parametro, e i due registri *A0* e *D0* conterranno, nell'ordine, l'indirizzo ove reperirla e la sua lunghezza (che comprende anche il Return finale).

In pratica, per gestire il parametro, il programma non dovrà fare altro, come prima operazione, che recuperare il contenuto di *A0* e *D0* e salvarlo in una costante o nello stack.

Vediamolo con un esempio, che svolge proprio il compito cui il lettore fa riferimento: preleva la stringa parametro, e la stampa sul video.

In pratica, un vero e proprio *Print*, molto stringato, ma efficace e veloce, forse più del tradizionale *Echo*.

Si copi, dunque, il disassemblato presente nel riquadro con l'editor del *Seka Assembler* (per entrarvi, tasto *Escape* dopo aver lanciato il programma).

Si torni poi al prompt *Seka>* ripremendo *Escape*, e si impartisca il comando **W** per salvare su periferica il sorgente (non si sa mai...).

Si assembli la routine con **A** (e magari **H** come opzione quando richiesto) e, se non si sono verificati errori, si proceda infine con *WO*, che produrrà il file eseguibile cui assegneremo il nome *Print* (o altro).

Per testarne il funzionamento, si impartisca in una finestra Shell oppure Cli...

Print quello che vuoi

...seguendo le consuete regole del Dos, ovvero segnalando il path completo del comando se si trova in una directory o disco diversi da quella corrente (p.es. *df1:Print...*).

Come risultato, vedremo stampare su schermo la stringa *"quello che vuoi"*.

La routine, come si può notare, è estremamente semplice (anche se non molto originale), e può essere ampiamente migliorata.

Non fa altro che inserire nello stack, come primissima operazione, i registri A0 e D0, per evitare che il loro contenuto venga perso nelle successive fasi, dato che questi registri vengono comunemente adoperati dalle funzioni di libreria.

Volendo, si potrebbe decurtare la lunghezza (in D0) di una unità, eliminando così il carattere di ritorno carrello che è incluso nella stringa.

Non è invece necessario porre il classico **0** in coda, dato che alla funzione *Write* viene comunicata la lunghezza della stringa.

Le specifiche di quest'ultima (A0 e D0) vengono prelevate dallo stack dopo l'apertura della *Dos Library* e il e la determinazione dell'*handle* di output, necessari per attivare la funzione *Write*.

Quest'ultima richiede, prima di essere richiamata dal tradizionale *Jsr*, che in D1 sia presente l'Handle di output, in D2 l'indirizzo della stringa e in D3 la sua lunghezza

```
Sysbase: equ 4
LVOopenlibrary: equ -552
LVOcloselibrary: equ -414
LVOoutput: equ -60
LVOwrite: equ -48
* salva indirizzo e lunghezza
 movem.l a0/d0, -(sp)
* apre libreria dos
 move.l #nomlib, a1
 moveq #0, d0
 move.l sysbase, a6
 jsr LVOopenlibrary (a6)
 tst.l d0 ; se 0, errore
 beq uscita
 move.l d0, a6 ; base doslib.
* preleva handle output
 jsr LVOoutput (a6)
 move.l d0, d1 ; per write
* scrive stringa parametro
 movem.l (sp) + , a0/d0
 move.l a0,d2
 move.l d0, d3
 jsr LVOwrite (a6)
* chiude libreria dos
 move.l a6,a1
 move.l sysbase, a6
 jsr LVOcloselibrary (a6)
*
uscita: rts
*
 nomlib: dc 'dos.library', 0
```

vrebbe installare la *Recoverable Ram* si blocchi all'inizio, ovvero quando tenta di creare la directory **L** nella *Rad*, fa pensare ad un qualche errore di copiatura.

Con molta probabilità, da qualche parte si è adoperato *Ram:* invece di *Rad:*.

Il tre file batch pubblicati nell'articolo, comunque, sono anche stati inseriti nel disco *Amigazzetta 6*, e funzionano correttamente.

Per consolarti, prova a ricordare il primo mese di contatto con il C64: funzionava tutto al primo tentativo?

CAMPUS
AMIGA

66 - LADRI DI MUSICHE

Lavorando con Amiga capita spesso (sempre?) di ascoltare le meravigliose musiche che fanno da sottofondo a videogames o programmi demo di varia natura. Non tutti sono abili compositori di brani musicali, e nemmeno numerosi sono coloro in grado di usare i tanti programmi in grado di gestire la musica. Ecco, dunque, la possibiltà, operando in *Seka Assembler*, di "estrarre" i brani musicali, dai programmi che possediamo, per inserirli in nostri listati. Inutile dire che la procedura è consigliata a chi conosce abbastanza bene Amiga e, soprattutto, ha una certa dimestichezza con la gestione di un programma assemblatore.

74 - OBJECT A BORDO!

Al contrario del programma precedente, questo listato consente anche ai principianti (dotati, però, di buona volontà) di realizzare una procedura decisamente interessante, ed altrettanto "universale", per la gestione semplificata di *oggetti grafici*. Si tratta, in pratica, di un "Data Maker" (di 64-iana memoria) in grado di far incorporare a nostri programmi Basic i dati relativi a Sprite e Bob realizzati comodamente, a parte, con uno dei tantissimi *tool* di pubblico dominio. Un mini gioco dimostrativo consente al lettore di meglio comprendere la tecnica descritta.

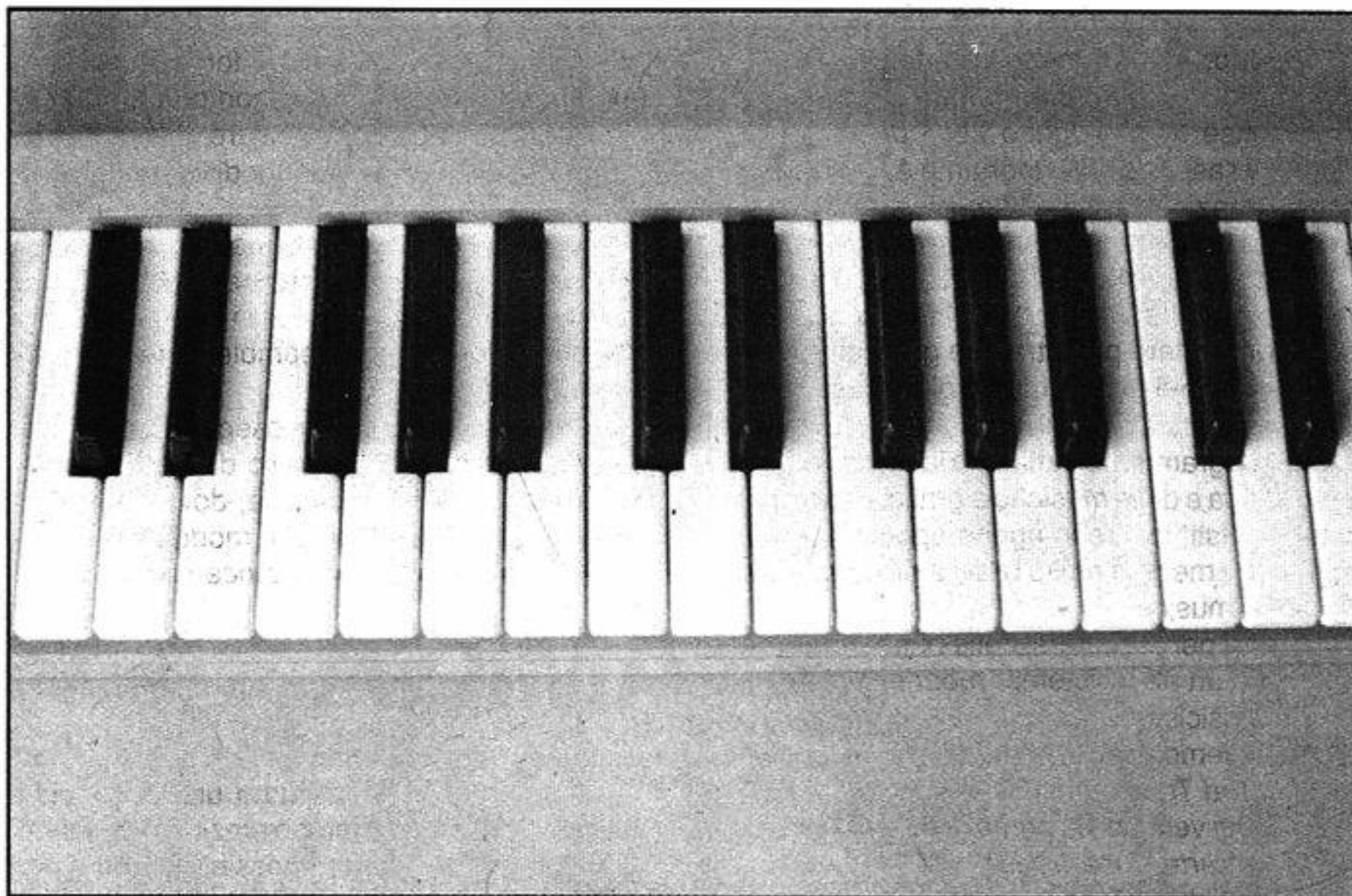
Nessun albero viene abbattuto per gli inserti di *Commodore Computer Club*, stampati su carta riciclata al 100%

LE AVVENTURE DI

PRIMO GIOVEDINI

Gara di acrobazia (4° file)

Finalmente l'attesa competizione di acrobazia ha inizio.
Le squadriglie delle due porta=aerei sono in volo ed i piloti dell'A.L.U., avversari di Giovedini &C=, iniziano per primi l'esibi=zione...
L'incrociatore AEGIS controlla con cura i Byte del punteggio...

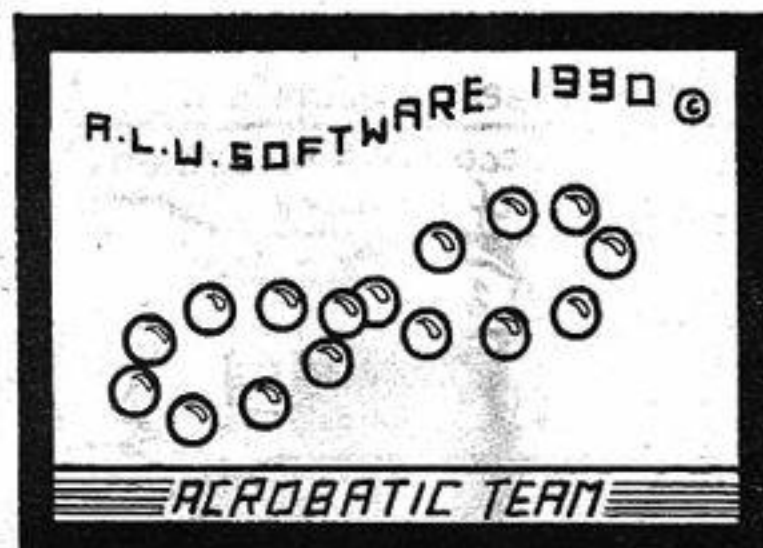
Le musiche che spesso corredano i programmi posti in commercio funzionano "sotto" Interrupt, vale a dire senza disturbare il programma principale

LADRI DI MUSICHE

Una procedura efficace per impadronirsi delle musiche dei vostri videogames preferiti; e per imparare l'Assembly di Amiga

di Donato De Luca

Comporre musica è più difficile che programmare in un qualsiasi linguaggio di programmazione. Chiunque può imparare a programmare, più o meno bene, a seconda del tempo che vi dedica e a seconda della sua predisposizione. Per imparare a programmare non ci vogliono particolari doti oltre ad una spiccata tendenza al masochismo spirituale e materiale.

Le cose appena dette non valgono, però, per imparare a comporre musica o a disegnare: se non si ha un po' di talento artistico non si riuscirà mai a comporre musica decente, che non sembri il lamento di un cane, oppure a fare un disegno che non sembri un quadro astratto.

Tuttavia un gioco, privo di una buona grafica o una buona musica, rischia di sfigurare rispetto

ad un altro, magari meno divertente e meno complesso.

Le stesse cose valgono per i *demo* in cui, anche se l'aspetto più importante è quello della programmazione, anche la musica e la grafica ricoprono un ruolo molto importante.

Che cosa possiamo fare allora, noi poveri programmatori senza un briciolo di talento artistico, per rendere più attraente graficamente e musicalmente il nostro ultimo gioco con scrolling parallattico a 6 livelli?

Molti programmatori affidano la realizzazione della grafica e delle musiche a grafici e musicisti professionisti; inoltre in ogni gruppo di pirati vi è uno o più membri molto bravi a disegnare e a comporre musica.

Tuttavia, per procurarsi delle buone musiche vi è anche un altro sistema: *rubarle*.

Molti musicisti per comporre le musiche destinate ai demo usano un editor musicale chiamato *Sound Tracker*, di cui ne sono state fatte un'infinità di versioni, la prima delle quali è stata scritta da *Karten Obaraski*.

Tali musiche possono essere inserite con elevata facilità nei giochi, nei demo, perfino nei word processor, poichè la stragande maggioranza delle routine replay (che, ricordiamo sono quelle routine che hanno il compito di suonare le musiche) girano in *interrupt*, di solito durante quello generato dal *Vertical Blank*.

Dal momento che le musiche seguono sempre la stessa struttura, dal punto di vista dell'organizzazione dei dati, qualche furbacchione ha pensato che, una volta che il demo ha finito di essere eseguito, i dati della musica saranno ancora presenti in memoria.

Conoscendo la struttura con cui viene salvato un *modulo* (ricordiamo che un modulo contiene i dati riguardanti la musica realizzata con il sound tracker e anche quelli relativi agli strumenti utilizzati), basterà scandagliare l'intera memoria alla ricerca della fatidica struttura dati; una volta trovata la si potrà salvare.

Questo è il principio su cui si basano i *Sound Rippers*. Essi in genere, una volta localizzato il brano musicale, possono salvare i suoi dati, e quindi la musica stessa sotto forma di modulo o di musica (la quale, però, non contiene i dati degli strumenti utilizzati), salvare solo uno strumento, suonare la musica, modificarne il nome ecc..

Tuttavia, per far suonare una musica salvata sotto forma di modulo, ci deve essere *qualcosa* che la suona. Questo qualcosa non è altro che una breve routine in assembler chiamata, di solito, *Replay Routine*.

Ciò implica che, per far eseguire la musica appena rubata senza l'ausilio di un sound-ripper o di un editor compatibile, dovremo usare per forza un assemblatore in modo da unire in matrimonio la routine suonatrice con i dati da suonare.

RIPPER TIME

Recentemente ci è capitato un *Ripper* che include, tra le sue numerose funzioni, un'opzione mai apparsa prima: la possibilità di salvare la musica direttamente come file eseguibile, per farla successivamente suonare senza *passare* per l'assemblatore; il che, obiettivamente, è una pacchia.

Il suddetto programma si chiama *Soundtracker Utilitie* ed è un programma di pubblico dominio (freeware) scritto da *Gigicar* del gruppo *Phoenix* (che, se non andiamo errato è un gruppo di programmatori Belgi).

L'utility è strutturata in 5 menu: *Load, Save, Options, Help, Quit*. Inoltre è presente (in basso) un simpatico equalizzatore che monitorizza i quattro canali audio. Vediamo le funzioni principali:

Load: Offre tre opzioni, la prima delle quali permette di caricare un modulo da disco. Ricordiamo che un modulo, dato che oltre ai dati della musica contiene anche quelli degli strumenti utilizzati, spesso raggiunge i 60 kbytes, ma a volte supera anche i 100kbytes. L'opzione è molto comoda in quanto permette di rendere eseguibili nostre musiche composte, sempre se ne siamo capaci.

La ricchezza e la complessità delle opzioni disponibili non sempre esaudiscono le esigenze dell'utenza

COPIALO PER TELEFONO
Anche i listati presenti in queste pagine possono esser tirati giù per mezzo del modem; se, ovviamente, ne possedete uno.
La procedura per collegarsi con la nostra banca dati (attiva 24 ore su 24) è riportata su altra parte della rivista.
Chi non possiede il modem (che aspettate a procurarvelo?) può tuttavia richiedere il dischetto, contenente il software, presso il nostro servizio arretrati.

Save: Questo menu è la perla del programma.Contiene 6 opzioni: *Modules, Song, Sample, Executable, Plst, Quit.*

La prima opzione *(modules)* permette di salvare la musica trovata come modulo. Bisogna stare attenti allo spazio perchè un modulo ne occupa parecchio.

La seconda opzione permette di salvare solo i dati relativi alla musica.

La terza opzione permette di salvare uno o più strumenti.

La quarta opzione è quella che rende unico il programma perchè consente di salvare la musica trovata in memoria come un programma eseguibile direttamente da *Cli*, in quanto i dati riguardanti sia la musica che gli strumenti vengono salvati insieme ad una routine di replay.

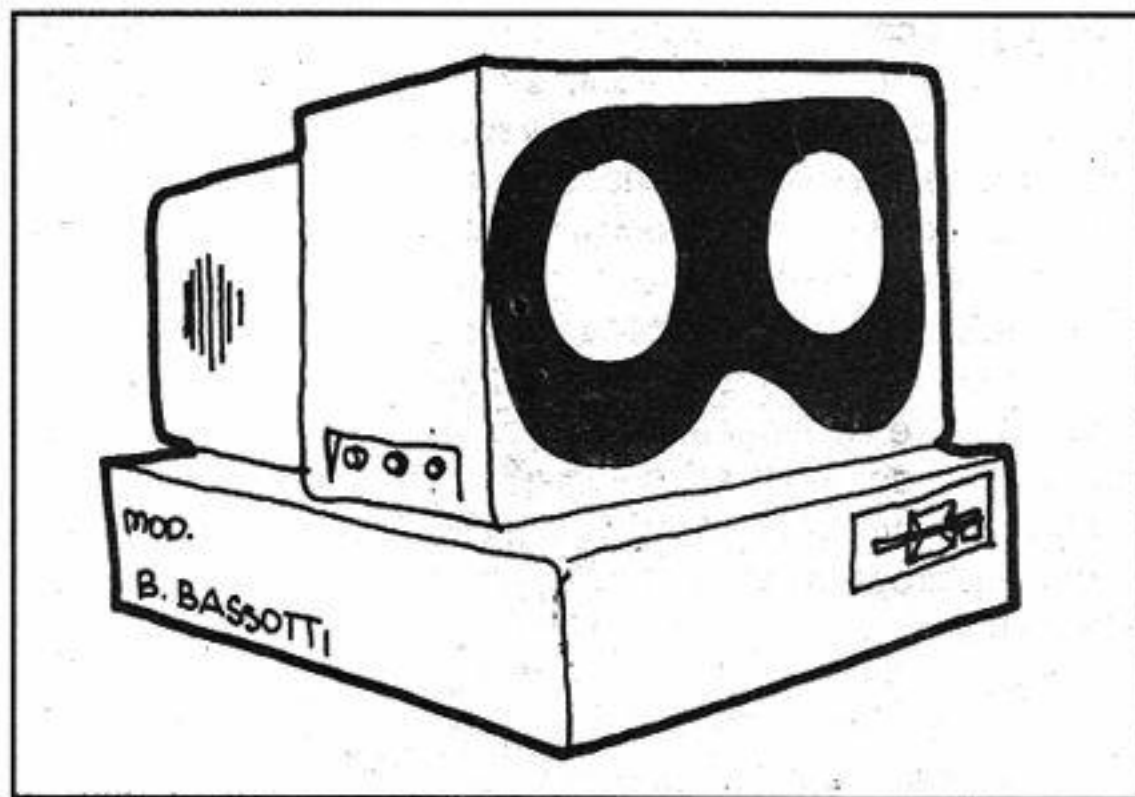
Purtroppo la routine ha tre *nei*. Il primo, al quale non possiamo porre rimedio facilmente, consiste nel fatto che non si può far eseguire il file ottenuto direttamente da WorkBench poichè alla routine di replay manca quella parte di codice che chiede il permesso al WorkBench di partire.

Il suo autore ha però affermato che rimedierà all'inconveniente in una prossima versione dell'utility. Al secondo neo, invece (ammesso che voi lo considerate tale), possiamo porre rimedio molto facilmente.

La routine smette di eseguire la musica quando viene premuto il pulsante sinistro del mouse; il che va benissimo in nostri programmi (purchè teniamo conto di tale particolare); se, però, volete inserire una musica (ricordiamo che la routine di replay gira in interrupt) in un programma che fa anch'esso uso del pulsante sinistro del mouse, la cosa seccherebbe un pò, per ovvi motivi.

Cambiando, però, un byte nel programma eseguibile, si potrà interrompere l'esecuzione tramite, ad esempio, la pressione del pulsante fire della porta 2. Ad ecceziode di alcuni giochi (e neanche poi tutti) sono ben pochi i programmi che fanno uso del pulsante di fuoco della porta due!

Per chi stesse meditando sull'opportunità di inserire una colonna sonora di propria composizione in un videogame commerciale, ricordiamo che, anche se la routine di replay gira in interrupt, è sufficiente che sia disabilitato l'interrupt di livello 6 (quello che viene dalla CIA) e la nostra musica non suonerà affatto.

Una caratteristica interessante della routine di replay utilizzata, è che essa ha una pelle molto dura perchè gira a suon di interrupt di livello 6, il quale non viene soppresso, al contrario di quello che accade con l'interrupt di

livello 3 (generato all'inizio di ogni Vertical-Blank, per chi se ne fosse dimenticato).

Successivamente sveleremo i segreti di alta chirurgia necessari per eseguire l'operazione sul file eseguibile.

Passiamo ora al menu **OPTIONS** che consente di effettuare 5 scelte:

1) *Change Names:* consente di cambiare il nome della musica e degli strumenti da essa utilizzati.

2) *Play Sample:* consente di suonare uno strumento

3) *Filter On / Off:* attiva o disattiva il filtro passa alto. Ricordiamo che l'Amiga 1000, e alcune serie di Amiga 2000 e Amiga 500, non dispongono della possibilità di comandare via software l'attivazione o la disattivazione di tale filtro.

4) *Play On / Off:* consente di suonare l'eventuale modulo trovato in memoria. Quando la musica viene eseguita, viene attivato in basso un equalizzatore a suon di Copper che monitorizza i 4 canali audio. Ovviamente, se non è stata trovato alcun modulo in memoria, questa opzione non darà alcun effetto.

5) *Quit:* ci sfugge il significato di tale inconsueta opzione...

L'opzione *Help* presente nel menu principale, come è facilmente intuibile, fornisce le istruzioni.

Il programma analizzato, pur essendo un ottimo programma ha, a nostro avviso, un grosso difetto, che colpisce specialmente gli utenti cui è destinato, cioè quelli meno esperti, che di solito hanno solo 512k oppure il vecchio Agnus (quello che riconosce come Chip-Ram solo i primi 512 kbytes di memoria). Indovinate qual è? L'occupazione di memoria!

Il programmino occupa circa 30k, quantità che può far sorridere chi dispone di 8 MegaByte di memoria. C'è da tener presente, tuttavia, che il programma possiede anche un 'interfaccia grafica la quale, vi assicuriamo, va tutta in chip-ram e non, purtroppo, negli altri 7 e passa mega, magari liberi.

VIETATO AI MINORI

E' probabile che i neo-utenti di computer non riescano a ben comprendere l'utilità della procedura software descritta nelle presenti pagine. Questa è infatti destinata a coloro che sono già padroni delle principali tecniche di programmazione e desiderano approfondire il modo in cui un elaboratore organizza i vari dati all'interno della memoria.
Il programma, pertanto, rappresenta un invito ai lettori più esperti, soprattutto a coloro che, considerando il listato come una base di partenza, riescano a pervenire a procedure più interessanti e capaci, magari, di offrire applicazioni di più ampio respiro.
Se hai incominciato da poco, comunque, non scoraggiarti! Tutti coloro che, oggi, vantano una particolare competenza nel campo dell'informatica hanno iniziato con un banale Print "Pippo". Perchè non dovresti riuscire anche tu?...

Vi è quindi il rischio concreto che l'interfaccia grafica del programma si sovrapponga ai dati della musica.

La situazione peggiora se si hanno solo 512k perchè, oltre al pericolo precedente, anche il Ripper si potrà sovrapporre ai dati.

ALTA CHIRURGIA

Prima di descrivere l'operazione vera e propria, documentiamoci come farebbe ogni bravo chirurgo.

Disassemblando un modulo eseguibile prodotto da Ripper (di cui se ne può vedere uno stralcio in figura 1) si vede che vi è l'istruzione...

btst #6, $bfe001

...la quale, lo diciamo per l'ennesima volta, non fa altro che testare lo stato del pulsante fire della porta joystick n. 1 il quale corrisponde, appunto, al pulsante sinistro del mouse.

Tutti sappiamo che il bit 7 del medesimo indirizzo corrisponde al pulsante fire della porta joy n. 2. Basterà sostituire 6 con 7 e la replay

Non ci stancheremo mai di ripetere che è indispensabile eseguire i vari esperimenti su dischi - copia

La procedura descritta può sembrare complessa, ma basta seguire le istruzioni per avere risultati soddisfacenti

routine cesserà di suonare la musica solo quando verrà premuto il pulsante fire della porta 2.

Per eseguire l'operazione ci serve un *monitor*, anche se l'optimum sarebbe un assemblatore con funzioni di monitor e dubugger (il *Seka* va benessimo). Prima di iniziare l'operazione trascrivete, a parte, la lunghezza (in bytes) esatta del paziente, cioè del file cui fare l'intervento.

1) Caricate il Seka

2) Scrivete **RI**. Alla richiesta *FileName* date il nome del file esegubile cui volete fare l'operazione.

3) Alla richiesta *Begin*, scrivete $20000. Nota: potete scegliere anche un'altra locazione a partire dalla quale far caricare il file, ma state attenti a quello che fate perchè è cattiva educazione trasferire 60 / 100 kbytes da disco in ram senza avvertire Mr *Exec*! Per sbaglio potreste caricare il vostro blocco di dati (abbastanza cicciottello) in un posto non molto indicato: per esempio proprio dove è stato caricato il Seka o il monitor che state utilizzando.

In genere, disponendo di 1,5 Mega di ram, la zona di memoria tra $20000 e $70000 è libera, ma se volete fare un lavoro da persone bene educate, vi consigliamo di chiedere a Mr. Exec un blocco di memoria delle dimensioni dell'eseguibile che volete caricare. Se vi è memoria libera *Exec* darà l'indirizzo del blocco e voi, da scolari diligenti, lo annoterete da qualche parte per poi caricarvi il blocco dati.

Chi non conosce l'assembler del 68000 deve limitarsi a digitare il programmino di queste pagine (scritto in assembler, ovviamente), assemblarlo e mandarlo in esecuzione.

Una volta eseguito il programma (a meno che non abbiate commesso errori) avrete in D3 un valore.

Se questo è $00000000 vuol dire che non c'è memoria libera a disposizione: comprate quindi un'espansione, oppure (nell'ordine) vi buttate dalla finestra o restituite la memoria utilizzata a Exec.

Se, invece, il valore è diverso da $00000000, allora caricate a partite da tale valore il blocco di dati.

Nel listato, al posto di *Francy*, digitate la lunghezza (in Bytes) del blocco dati che volete caricare. Come i più avranno capito, il programmino non può essere di alcun aiuto se non lo facciamo eseguire utilizzando un *debugger:* il risultato viene fornito in un registro. Tuttavia visto che siamo bravi, abbiamo pensato anche a quei poveretti che possiedono solo un assemblatore (o che non lo possiedono e si sono assemblati il codice a mano); *AskMemCli* (come dice il nome) può essere eseguito anche da CLI poichè scrive nella finestra corrente il risultato.

```
;Ask.mem
;Sinstassi Seka
 AllocMem = -198
ExecBase = $4
Move.l Execbase,a6
Move.l #Francy,d0
Move.l #0, d1
Jsr AllocMem(A6)
Move.l d0, d3
Rts
```

Il miniprogramma in sintassi Seka

Ignorate la richiesta successiva, semplicemente tramite la pressione del tasto return.

4) Adesso andate alla locazione $20000 + $CB (o quella che avete scelto voi + $CB).

5) Modificate **06** con **07**.

6) Scrivete **WI**.

7) Alla richiesta *Begin* scrivete *$20000*, o quello che avete scelto voi.

8) Alla richiesta *End* scrivete $20000 + (lunghezza -1), dove lunghezza è la lunghezza del paziente.

9) Uscite dal Seka e provate a vedere se il programma funziona.

COMPATIBILITA'

Come abbiamo detto prima la routine funziona a suon di interrupt di livello 6, cioè quello che viene generato da uno dei 2 CIA, il quale ha una priorità più elevata rispetto a quello generato ad ogni vertical-blank che ha priorità 3.

Tuttavia l'accorgimento non dona di certo la vita eterna alla routine di replay, in particolar

modo è sempre meglio evitare che questa giri quando l'Amiga è molto indaffarata, poichè si potrebbero causare serii problemi di sincronizzazione (e *Guru* conseguenti).

Evitate, poi, di far funzionare la routine mentre sta lavorando un interprete o un compilatore; se proprio volete inserire la musica nei vostri giochi scritti in basic allora, tramite compilatore apposito, compilateli, dal momento che si dovrebbe avere una convivenza abbastanza pacifica. Ricordiamo infine che, a causa della struttura della replay routine utilizzata, vi è una serie di istruzioni eseguite fuori dal ciclo di interrupt. Queste ci constringono, nel caso vogliamo la musica contemporaneamente a un altro programma, a far eseguire il brano tramite il comando Cli: Run.

Esempio: supponiamo che il file *musica.game* sia l'eseguibile prodotto dal ripper e che *Gioco* sia l'eseguibile prodotto da un compilatore (basic, c, modula, o quel che vi pare).

Nella *startup-sequence* dovrete inserire...

Run musica.game Gioco

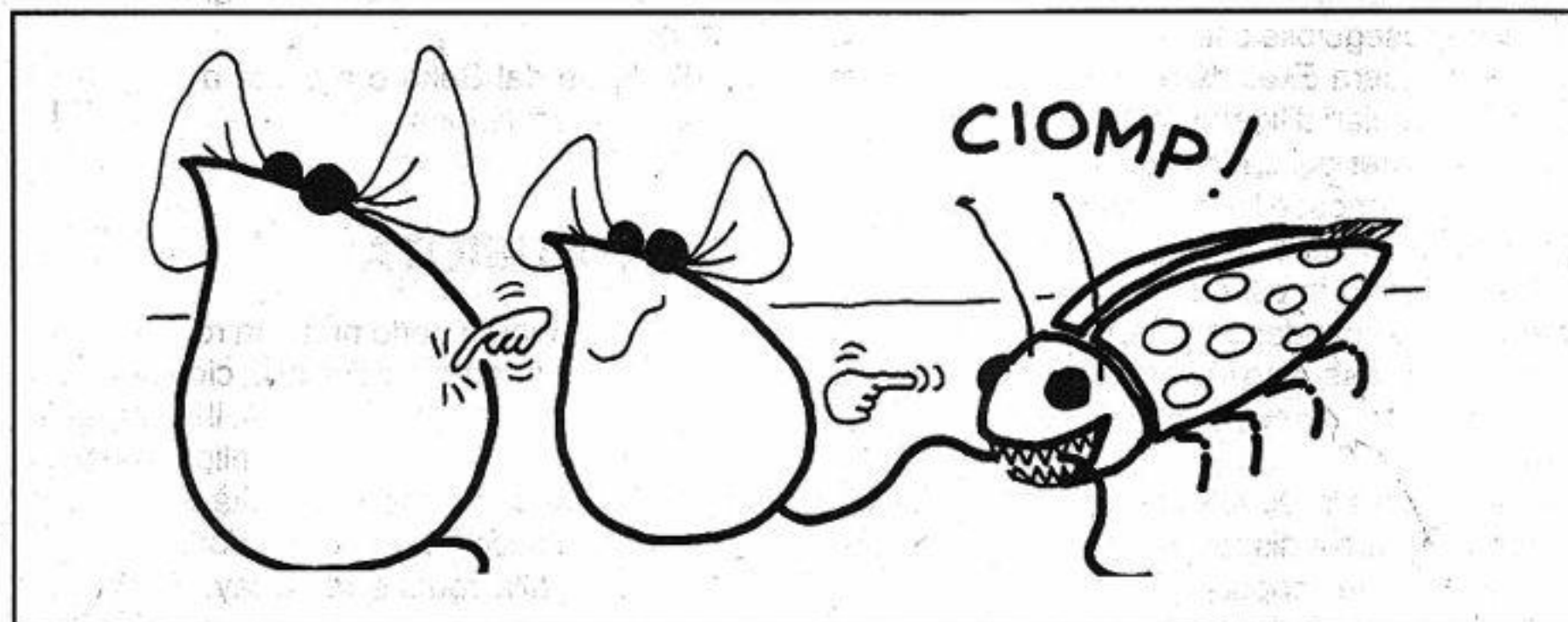
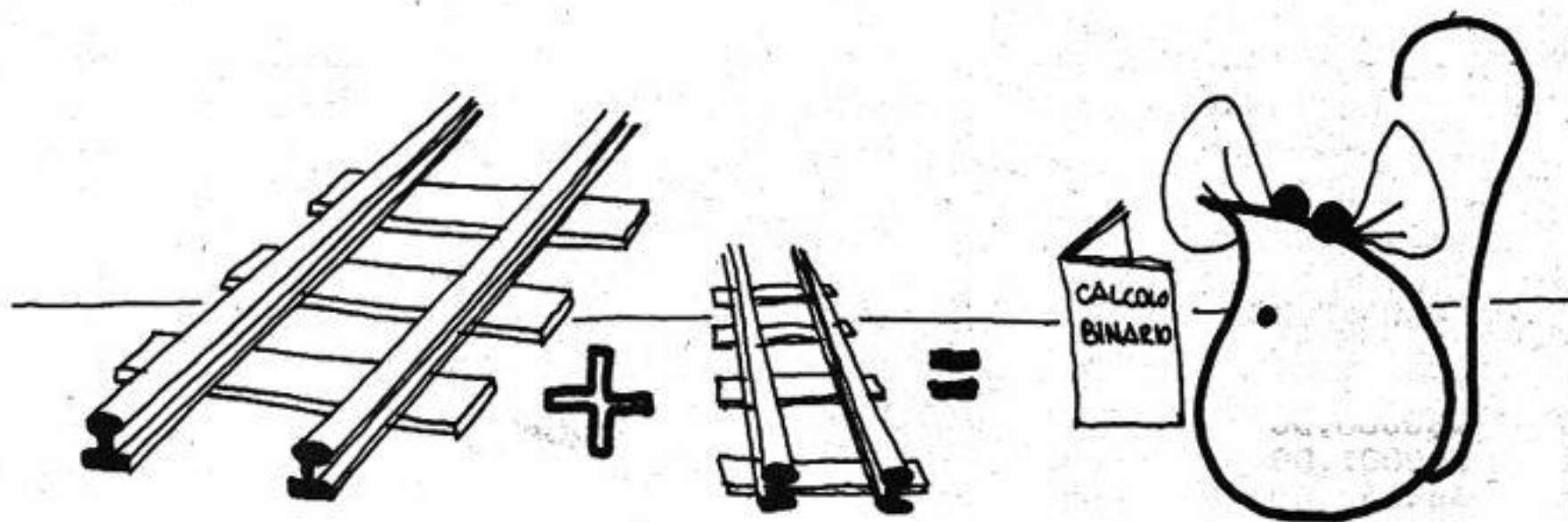
E' da notare che chi ha solo 512 kbytes avrà serii problemi, derivanti da due fattori. Con 512 kbytes non è molto consigliabile spendere 100 kbytes per una sola musica! Inoltre (e qui si dovrebbe strappare a morsi le orecchie del creatore della routine di replay), quest'ultima trasferisce forzamente tutti i dati della musica a partire da $30000; vi lasciamo immaginare che cosa può succedere se a $30000 c'era qualcos'altro, per esempio un programma o i suoi dati.

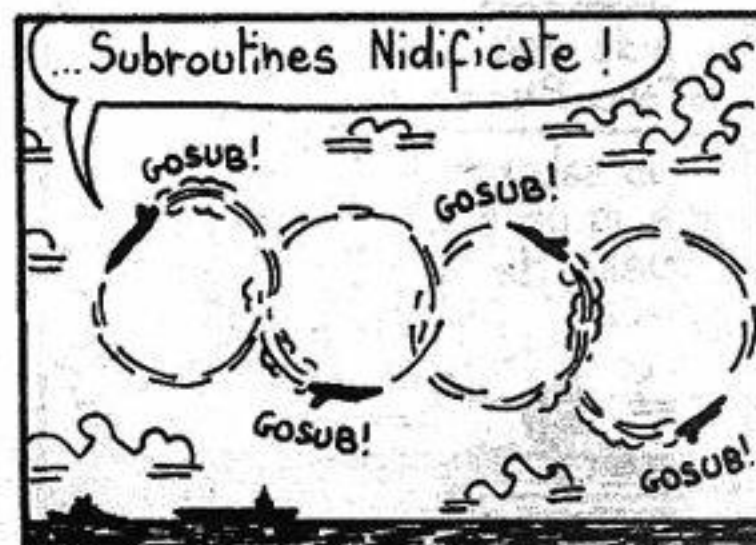
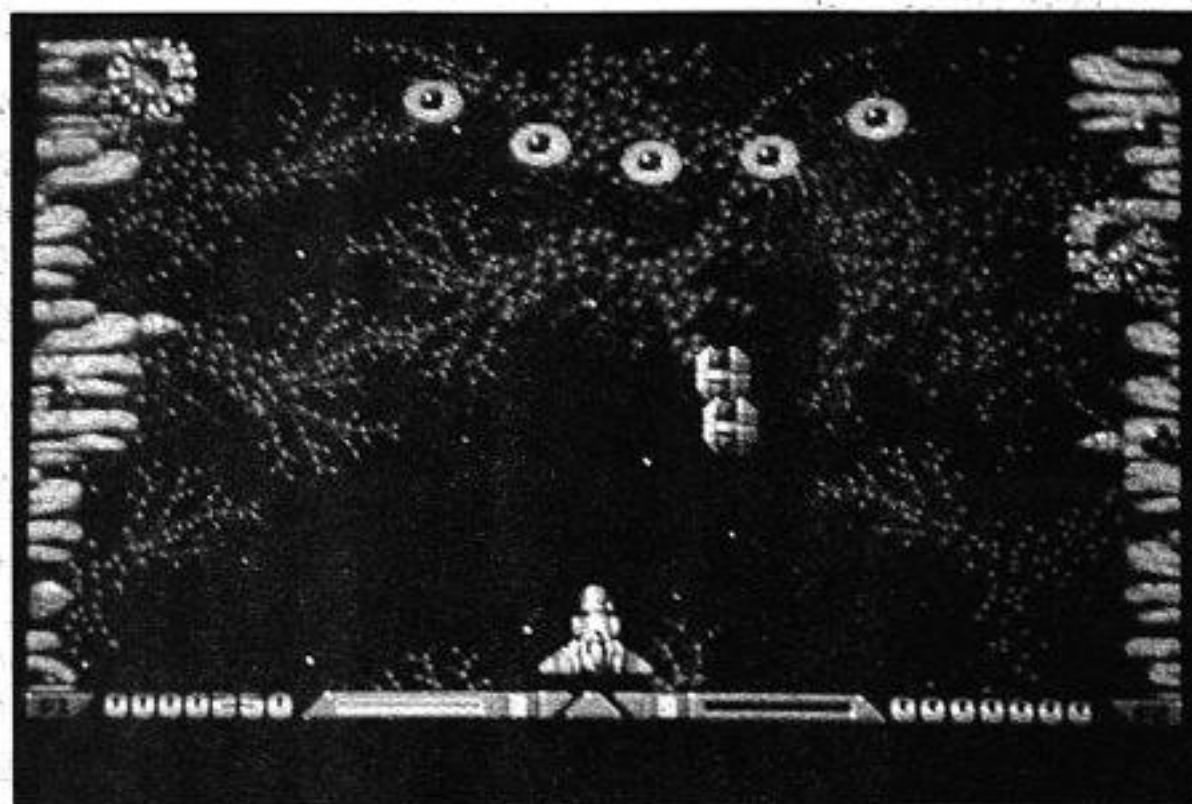
Potevano certo fare a meno di realizzare una procedura del genere in un programma destinato ad un pubblico formato, per lo più, da utenti poco esperti; quelli più esperti, infatti, non usano questo ripper, per altro limitato.

ASKMEMCLI

Il programma (più lungo) presente in queste pagine (*Askmemcli*, appunto) non fa altro che chiedere a *Exec* un blocco di memoria tramite *AllocMem;* succesivamente converte l'indirizzo di tale blocco in codici Ascii tramite un algoritmo di conversione (che non è il massimo dell'efficenza, però funziona) e infine fornisce nella finestra corrente *(Cli)* tale indirizzo.

Va detto che se il programma, per qualche motivo, non riesce ad aprire la *dos.library*, non verrà stampato niente; inoltre, dato che il programma ovviamente non restituisce la memoria presa (se no, che scopo avrebbe?) evitate di farlo girare più di una volta di seguito, altrimenti vi renderete conto che anche 2 mega di ram sono pochini...

La disponibilità di grandi quantità di memoria Ram può facilitare lo svolgimento delle operazioni

```
;  ASKMEMCLI
;
;DONATO DE LUCA
;

EXECBASE     = $4
ALLOCMEM     = -198
OPENLIBRARY  = -552
CLOSELIBRARY = -414
FREEMEM      = -210
OUTPUT       = -60
WRITE        = -48

INIZO:

move.l EXECBASE,a6  ;Execbase a6
move.l #1000,d0 ;Quanti bytes?
;Mettete la lunghezza del file!!!!!!
move.l #0,d1      ;Di che tipo?
jsr ALLOCMEM(a6)     ;Li prendo.
move.l d0,valore     ;Salvo il risultato
clr.l d0             ;in d0. Pulisco d0

lea stringa,a1 ;Dove metto i
               ;codici ascii
lea valore,a0  ;Da dove prendo
               ;i nibble da
               ;converire
moveq #4,d4    ;Quanti bytes?

ConLOOP:       ;Converto
lea tab,a2     ;Tabella di conv.
move.b (a0),d0 ;Prendo 1 byte
lsr.b #4,d0    ;Prendo il nibble
               ;piu` significativo
add.l d0,a2    ;Ricavo codice ASCII
move.b (a2),(a1)+;Salvo codice ASCII
lea tab,a2     ;Tabella di conv.
move.b (a0),d0 ;Riprendo il byte
and.b #$0f,d0  ;Prendo il nibble
               ;meno significativo
add.l d0,a2    ;Ricavo codice ASCII
move.b (a2),(a1)+;Salvo codice ASCII
add.l #1,a0    ;Avanzo di un byte
dbra d4,CONLOOP;Per 4 byte (8 nibble)

comunica:             ;Apro la DOS library.
lea dos,a1            ;Testo d0 , se e` nullo
jsr OPENLIBRARY(a6);vuol dire che ci sono
;tst.l d0              ;problemi con l`apertura
;beq NONAPROLADOSLIBRARY;della DOS.LIBRARY
move.l d0,a5   ;Salvo l`indirizzo in A5
jsr OUTPUT(a5)     ;Cerco dove spedire i dati
move.l d0,d1       ;Dove li spedisco
move.l #stringa,d2;Da dove li prendo
moveq #8,d3        ;Quanti ne prendo
jsr WRITE(a5)      ;Li spedisco(buon viaggio)
move.l a5,a1       ;Chiudo la DOS.LIBRARY
jsr CLOSELIBRARY(a6);
rts                ;Torna dalla mamma , Lessie

NONAPROLADOSLIBRARY:
move.l VALORE,a1 ;Se non sono riuscito ad
move.l #1000,d0  ;aprire la dos.library rido
jsr FREEMEM(A6)  ;la memoria presa ad exec
rts              ;Pussa a via , Lessie
                 ;Ricordatevi di dare TUTTA

                 ;la memoria presa!!!!!!!!!
stringa:
dc.l 0
dc.l 0
valore:
dc.l 0
even
dos:
dc.b 'dos.library',0
tab:
dc.b `0`,`1`,`2`,`3`,`4`,`5`,`6`,`7`,`8`
dc.b `9`,`a`,`b`,`c`,`d`,`e`,`f`
```

```
 020000 OR.B     #$03F3,D0
020004 OR.B     #$0000,D0
020008 OR.B     #$0002,D0
02000C OR.B     #$0000,D0
020010 OR.B     #$0001,D0
020014 OR.B     #$44A3,D0
020018 OR.B     #$0001,D0
02001C OR.B     #$03E8,D0
020020 OR.B     #$44A3,D0
020024 MOVE.L   #$00011266,D0
02002A LEA      $02004A(PC),A2
02002E LEA      $030000.L,A0
020034 MOVE.L   A0,A1
020036 ADD.L    D0,A1
020038 ADD.L    D0,A2
02003A MOVE.B   -(A2),-(A1)
02003C CMP.L    A0,A1
02003E BNE      $02003A
020040 BSET     #$0001,$BFE001.L
020048 JMP      (A0)
02004A MOVEM.L D0/D1/D2/D3/D4/D5/D6/D7/A0/A1/A2/A3/A4/A5/A6,-(A7)
02004E BSR      $020058
020052 MOVEM.L (A7)+,D0/D1/D2/D3/D4/D5/D6/D7/A0/A1/A2/A3/A4/A5/A6
020056 RTS
020058 BSR      $0200F0
02005C CLR.B    $BFDE00.L
020062 MOVE.B   #$0082,$BFD400.L
02006A MOVE.B   #$0037,$BFD500.L
020072 MOVE.B   #$0081,$BFDD00.L
02007A MOVE.B   #$0011,$BFDE00.L
020082 MOVE.L   $000078.L,$0300A2.L
02008C MOVE.L   #$00030076,$000078.L
020096 MOVE.L   $0300A2.L,A0
02009C MOVE.L   $000078.L,A1
0200A2 CMP.L    $000078.L,A0
0200A8 BEQ      $0200B0
0200AC BRA      $020096
0200B0 BSR      $0201AE
0200B4 MOVE.L   $0300A2.L,$000078.L
0200BE RTS
0200C0 MOVEM.L D0/D1/D2/D3/D4/D5/D6/D7/A0/A1/A2/A3/A4/A5/A6,-(A7)
0200C4 BSR      $0201D0
0200C8 BTST     #$0006,$BFE001.L
0200D0 BNE      $0200D8
0200D4 BSR      $0200B0
0200D8 MOVE.B   $BFDD00.L,D0
0200DE MOVEM.L (A7)+,D0/D1/D2/D3/D4/D5/D6/D7/A0/A1/A2/A3/A4/A5/A6
0200E2 MOVE.W   #$2000,$DFF09C.L
```

La gestione degli sprite, con Amiga, richiede la loro presenza sul dischetto

OBJECT A BORDO!

Un datamaker su misura per rendere più comodo e trasportabile l'uso di Bob e Sprite. nel favoloso Amiga

di Domenico Pavone

Tra tutti i comandi implementati da AmigaBasic, il raggruppamento più consistente è quello riservato alla manipolazione dei cosiddetti "oggetti mobili", più comunemente intesi come *Sprite* e *Bob*.

Tale abbondanza, pur risentendo dei limiti qualitativi dell'interprete basic adottato dalla Commodore, di fatto consente una notevole facilità nel gestire le potenzialità grafiche di Amiga.

Lo scorrere sullo schermo di navicelle spaziali, proiettili perforanti e artiglieria varia, un tempo dominio esclusivo di programmatori provetti, è ora alla portata di tutti: qualche istruzione *Object.*, associata alla funzione *Collision*, ed il gioco (anzi, il Game) è fatto.

Non siamo però qui per approfondire gli aspetti teorici del problema, quanto per supplire ad una delle caratteristiche meno comode associate alla manipolazione di Sprite e Bob.

Scorrendo il manuale, infatti, si apprende con una certa soddisfazione che Amiga, o meglio AmigaBasic, non fa distinzione tra Bob e Sprite, quanto meno per ciò che riguarda la loro manipolazione.

La sola istruzione...

QUALCHE NOTA SUI LISTATI

Tanto il *Datamaker* che il *Demo* sono redatti in un basic che non richiede alcun approfondimento, se non la semplice consultazione del manuale per le istruzioni che eventualmente non si conoscono, e un'occhiata ai commenti annessi ai listati. Non viene adoperata alcuna libreria di sistema, proprio per favorire i meno esperti, che possono così cimentarsi nelle modifiche più svariate.

Del programma Datamaker vero e proprio, potrebbe essere migliorata la cosiddetta "interfaccia utente", ovvero semplificato e reso di maggiore impatto grafico l'input iniziale.

Ne conseguirebbe un inevitabile incremento delle dimensioni del listato, ma, a parte la maggiore comodità, costituirebbe un ottimo esercizio di programmazione. Per chi non ha voglia di pestare troppo sui tasti, la routine funziona egregiamente anche così com'è.

Si ricordi solo che, nei listati che dovranno gestire l'oggetto mobile, vanno sempre inserite le istruzioni di caricamento dei dati come nelle due righe del listato 3 delimitate dal commento *"legge dati creati col Dmaker"*.

Ovviamente, nel ciclo *For... Next* di lettura, il valore andrà modificato in accordo col numero dei Data, rilevabile dalla prima riga del file ottenuto con il Datamaker. Quest'ultima indicazione, una volta effettuato il *Merge*, può essere eliminata del tutto (è solo una Rem, in fondo).

Prima di concludere, val la pena notare come il programma Datamaker può trasformare in Data adoperabili da basic qualunque tipo di file, non solamente Bob e Sprite.

Potrebbe, per esempio, "processare" un normale file eseguibile; una volta ottenuto il file Ascii, basterebbe poi aggiungervi qualcosa come...

```
Open "nomefile" For Output As 1
For x = 1 To numerodati
Read a$: a% = Val (a$)
Print# 1, Chr$ (a%)
Next: Close 1: End
```

... per avere un classico "caricatore" basic, che ricrea, su disco, il file eseguibile. Unica accortezza, considerato l'algoritmo del Datamaker, il file da trattare non deve superare i 32767 byte, limite di capienza per la stringa (nel listato 1: oggetto$) che "ingloba" il file prima di trasformarlo in dati Ascii.

Una procedura che ricorda da vicino la tecnica usata con il C/64 per il disegno degli sprite

Object.Shape x, nome$

... renderà disponibile l'oggetto *Nome$*, cui viene assegnato il numero x, tanto che si tratti di uno Sprite quanto di un Bob.

Preso atto della difficoltà in meno da affrontare, vediamo quali problemi comporta la cosa.

Ancora sul manuale, viene riferito che alla variabile *nome$* sono associati gli attributi dell'oggetto: in parole povere, il disegno che lo compone, le sue dimensioni ed i suoi colori.

Tulle queste caratteristiche vengono automaticamente impostate adoperando il programma *ObjEdit*, presente nel disco *Extras* (directory *BasicDemos*). In pratica: prima si disegna l'oggetto con l'ObjEdit (o qualche *Sprite Maker* del pubblico dominio), poi lo si salva su disco ed infine, nel nostro programmino basic, va inserito un comando tipo...

Object.Shape 1, Input$ (Lof (1), 1)

... dopo aver adeguatamente aperto il relativo file (con *Open "nomeoggetto" For Input As 1*).

E fin qui, tranne le solite "traversie" per chi è alle prime armi, nulla di complicato.

Il programma basic che manipola l'oggetto, però, presenta l'inconveniente di dover essere sempre associato al file contenente i dati dello Sprite/Bob. Il che, volendo trasportare la routine su un altro dischetto, si traduce nella necessità

Sprite e bob sono due "oggetti" grafici simili tra loro solo in apparenza

di prestare particolare attenzione al *Path* (percorso) del file, pena la solita sfilza di "object not found".

TUTTO COMPRESO

Supponete, per esempio, di voler inviare un programma che manipola decine di sprite ad una rivista scelta a caso: CCC.

Il programma sarebbe facilmente testabile adoperando il dischetto che lo contiene, ma il suo listato non potrebbe *mai* essere pubblicato: senza i files che contengono gli sprite, non servirebbe proprio a niente.

Una soluzione radicale a tutti questi problemi, consiste nell'inserire i dati relativi agli oggetti nell'ambito del listato stesso, sotto forma di istruzioni Data.

D'altra parte, un Bob di dimensioni non microscopiche, magari composto da svariati bitplanes, può raggiungere facilmente una dimesione di migliaia di singoli dati. Improponibile, dunque, comporre da sè l'immagine, ammesso che si riesca a trovare il bandolo della matassa consultando il Rom Kernel Manual, non proprio semplice per chi è alle prime armi.

Ma, ed eccoci al dunque, è sempre possibile ricorrere all'ObjEdit per creare l'oggetto, salvarlo su disco, e solo successivamente trasformare in Data il file in questione.

DA FILE A FILE

Dopo aver attivato Amigabasic, si copi il listato, e lo si salvi debitamente su disco. Il programma è molto semplice da usare, e, entro certi limiti, rende il più facile possibile la vita a chi non si considera un esperto. Per verificarne e capirne in pratica il funzionamento, che come ovvio richiede la presenza di un oggetto già preesistente, sfrutteremo il file *Ball*, inserito nella directory *Basicdemos* del disco *Extras* fornito assieme al computer. Dopo il *Run* appare un primo input, che richiede il nome del disco e della eventuale directory ove ricercare il file da trattare. E' importante, nel fornire questo dato, attenersi alle comuni regole del Dos, inserendo debitamente il simbolo due punti(:) dopo il nome del disco o della periferica e, in presenza di una (o più) subdirectory, concludendo l'input con il simbolo di barra inclinata(/).

Nel nostro caso, ad esempio, bisognerà digitare...

Extras 1.3: BasicDemos/

... e Return per specificare il disco o la directory voluta.

Omettendo la risposta, verrà considerata la directory corrente, eventualmente modificata direttamente con il comando basic *Chdir*.

Chi possiede due drive, può ovviamente adoperare le comuni specifiche Df1:, Df0:, Ram:, eccetera. Viene poi richiesto il nome del file: per questa prima verifica, rispondiamo *Ball*.

Una ulteriore scelta riguarda il numero di dati che costituiranno ogni riga Data, valida più che altro a fini estetici, che è stata limitata tra 5 e 15, valori ottimali anche per differenti risoluzioni di schermo. Se si adopera la media risoluzione, 10 andrà più che bene. Infine occorre precisare *dove* dirigere l'output, con le stesse regole del Dos viste in precedenza. Anche qui il parametro può essere omesso, nel qual caso verrà assunta la directory corrente.

Si scelga il nome del disco (o della periferica) desiderato, che per il nostro primo esperimento potrebbe anche essere *Ram:* (e Return).

A questo punto, i dati immessi vengono mostrati sullo schermo, per verificare la presenza di eventuali errori. Prima di rispondere con *S* (sì) oppure *N* (no), si controlli attentamente il nome dei file di Input e Output, che devono comprendere l'intero percorso.

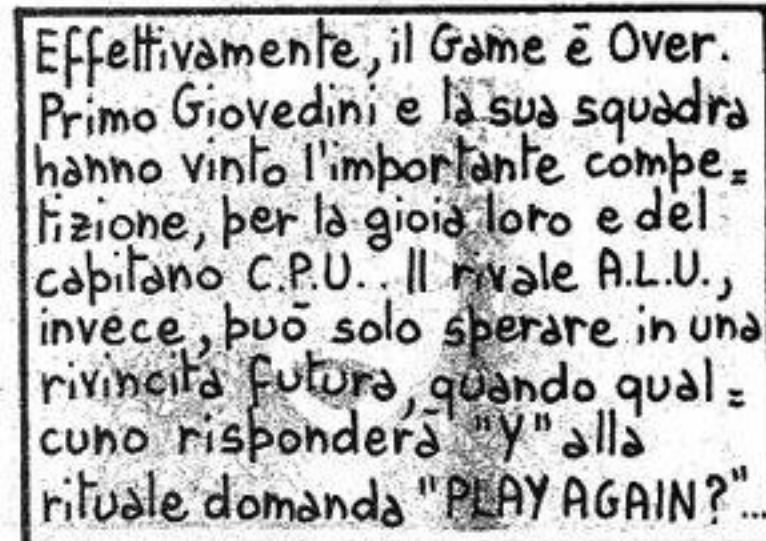
Il nome del file di Input, se si è seguito quanto finora descritto, dovrà risultare esattamente...

Extras 1.3: BasicDemos/Ball

...mentre quello di output assumerà il nome *Ram: Ball.data*

Il suffisso *".data"* viene automaticamente aggiunto dal programma al nome del file origine.

Se tutto è in regola (in caso contrario basterà reimmettere correttamente tutti i dati) e si è

Effettivamente, il Game è Over. Primo Giovedini e la sua squadra hanno vinto l'importante competizione, per la gioia loro e del capitano C.P.U.. Il rivale A.L.U., invece, può solo sperare in una rivincita futura, quando qualcuno risponderà "Y" alla rituale domanda "PLAY AGAIN?"...

BOB O SPRITE?

Il capitolo 7 del manuale del basic, illustra a grandi linee l'argomento *objects*, spiegando (succintamente) le differenze esistenti tra Sprite e Bob. Ma quando realmente conviene adoperare uno piuttosto che l'altro?

Il primo elemento che può determinare la scelta, è certo dovuto alle dimensioni possibili dell'oggetto. Adoperando uno sprite, come noto, non si possono superare i *sedici pixel* in orizzontale, il che limita alquanto le reali possibilità di questo tipo di oggetto.

I sedici pixel, si badi, sono da intendersi delle stesse dimensioni rilevabili su uno schermo 640 x 200 (il classico modo "80 colonne"), anche se si sta usando una risoluzione minore (320 x 200).

Per tale motivo gli Sprite vengono usati per lo più come puntatori (la freccetta del mouse è proprio uno sprite), o proiettili vaganti in games d'azione, grazie anche alla loro totale indipendenza dallo schermo sottostante.

Indipendenza dovuta alla presenza, tra i circuiti di Amiga, di un hardware espressamente dedicato alla loro gestione (il chip Daphne), motivo della maggiore velocità di movimento rispetto ai cosiddetti Bob.

Questi ultimi, però, sono in effetti gli oggetti più utilizzati, vuoi per le maggiori dimensioni possibili, che per il maggior numero di colori implementabili, negli sprite limitati a soli 3.

La loro gestione è affidata, in Amiga, al cosiddetto *Blitter* (Bob sta per Blitter OBject) che, durante l'animazione, provvede a ridisegnare l'oggetto nella nuova posizione e ripristinare il contenuto dello schermo sottostante. Per tale motivo, nonostante la capacità del Blitter di trasferire un milione di pixel per secondo, la velocità risulta inferiore a quella degli sprite. I Bob, in pratica, fanno parte dello schermo, nel senso che ne utilizzano lo stesso "background", anche se non in modo evidente.

I due tipi di oggetti, sono dotati delle stesse capacità di collisione, e relativa rilevazione, il che, in AmigaBasic, è collegato alla funzione *Collision*, ma con una grave lacuna: non esiste una rilevazione della collisione tra un oggetto ed un elemento dello schermo sottostante.

Sempre in ambiente AmigaBasic, c'è anche da notare una certa discontinuità nel movimento degli oggetti quando li si anima, dovuto all'uso da parte dell'interprete di suoi particolari interrupt, più "lenti" di quelli superveloci di Amiga.

Bob o Sprite, dunque? La risposta più corretta, è forse la più ovvia: dipende...

Nel dischetto Extras di Amiga sono presenti alcuni files utili per far funzionare la procedura di queste pagine

premuto il tasto N, la routine comincerà a svolgere il suo compito.

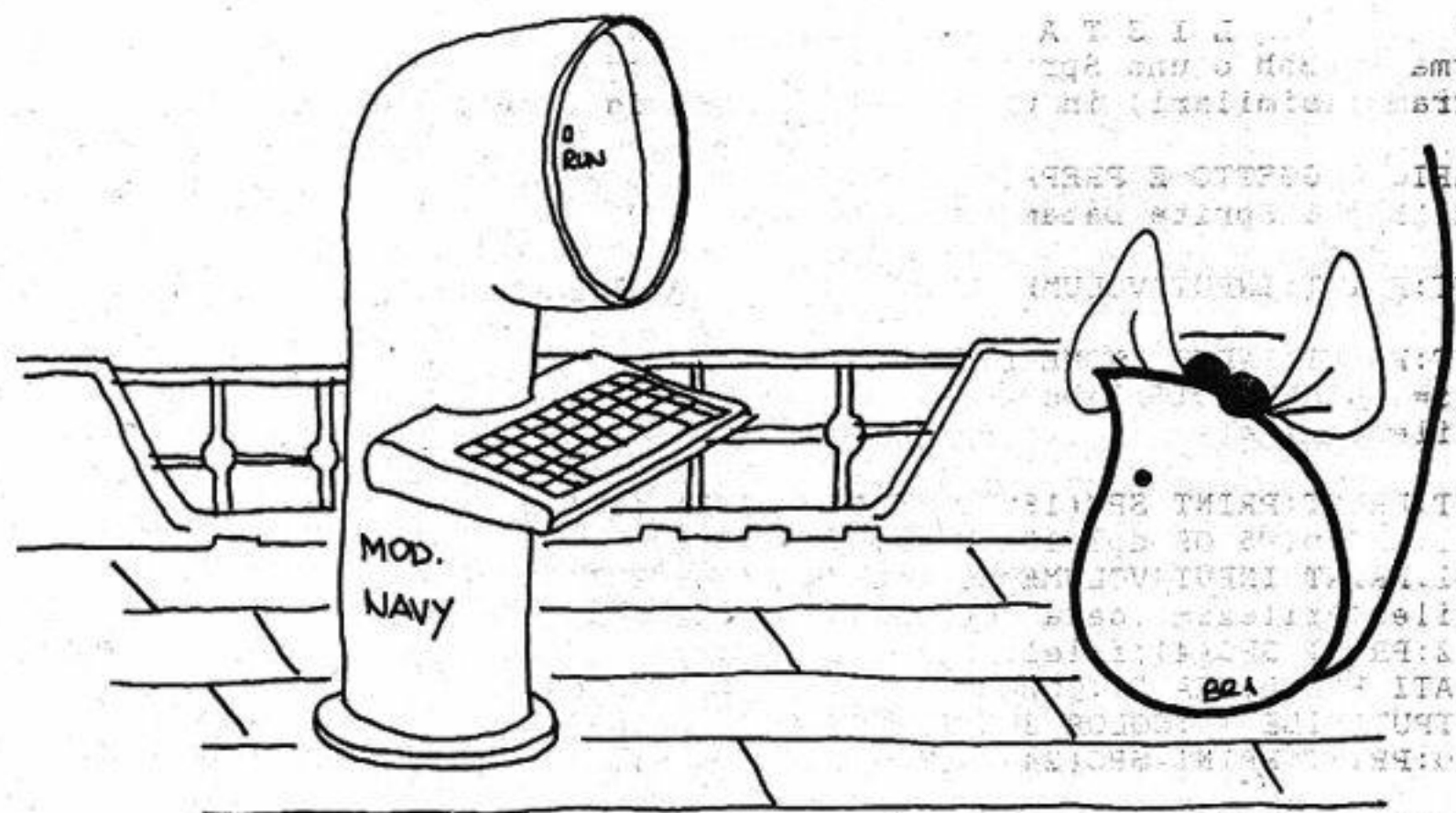
Eventuali inserimenti di dischetto verranno segnalati direttamente dal sistema: basterà, in questo caso, obbedire ed attendere qualche secondo, senza la necessità di clickare nel requester. Il frutto di tante fatiche (?) sarà un file Ascii di nome *Ball.data* presente, se così si è scelto, nella Ram Disk.

Per verificarlo, si può semplicemente caricarlo tramite l'editor del basic, oppure, dopo aver aperto una finestra Shell, impartendo:

Type Ram: Ball.data.

In entrambi i casi, se tutto ha funzionato a dovere, il file dovrebbe risultare come mostrato dal listato 2 di queste pagine. Si noti come, in testa al file, sia presente una riga di commento contenente l'indicazione del numero totale di dati; informazione, questa, basilare per una corretta assunzione degli stessi in una stringa che specifichi l'oggetto.

I singoli valori Data, inoltre, devono risultare ben allineati e disposti ognuno su tre colonne (anche l'occhio vuole la sua parte).

Un minigioco consente di verificare la corretta gestione dell'oggetto grafico

Il file *Ball*, come specificato in più punti del manuale, definisce un oggetto a forma di pallina. Ora, con il nostro *Ball.data*, è possibile ragruppare in un unico listato tanto un programma che necessita di un *Bob- pallina*, che i *dati* della pallina stessa.

Ovviamente, una tantum, sarà necessario operare un *Merge*.

Vediamo un esempio su come procedere, sfruttando il Demo proposto dal terzo listato.

FUSIONE CALDA

Si imparisca *New* al basic, quindi si copi il listato 3 e lo si salvi su disco a scanso di eventuali Guru. Con il programma ancora presente nella finestra *List*, si digiti nella finestra di output...

Merge "Ram: ball.data"

... se si era scelta la Ram Disk per contenere il file Ascii prodotto dal nostro *Data Maker* (in caso contrario, l'istruzione sarà:

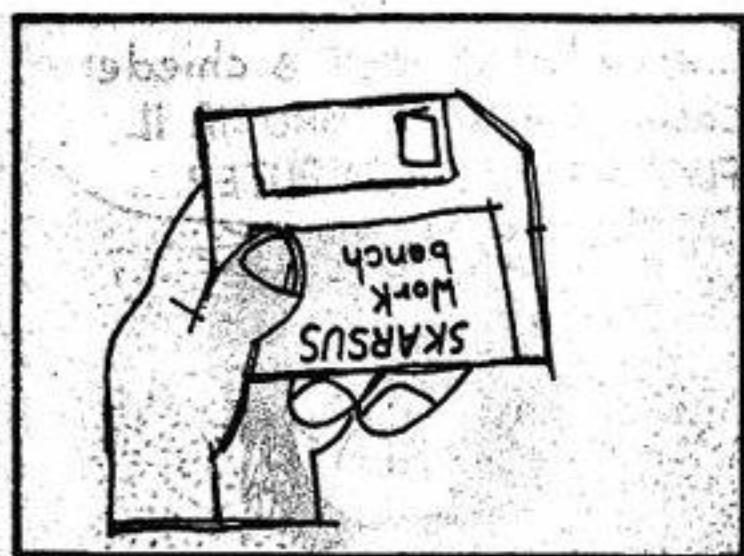
Merge "Nomedisco: ball.data"

Il programma Demo è ora completo.

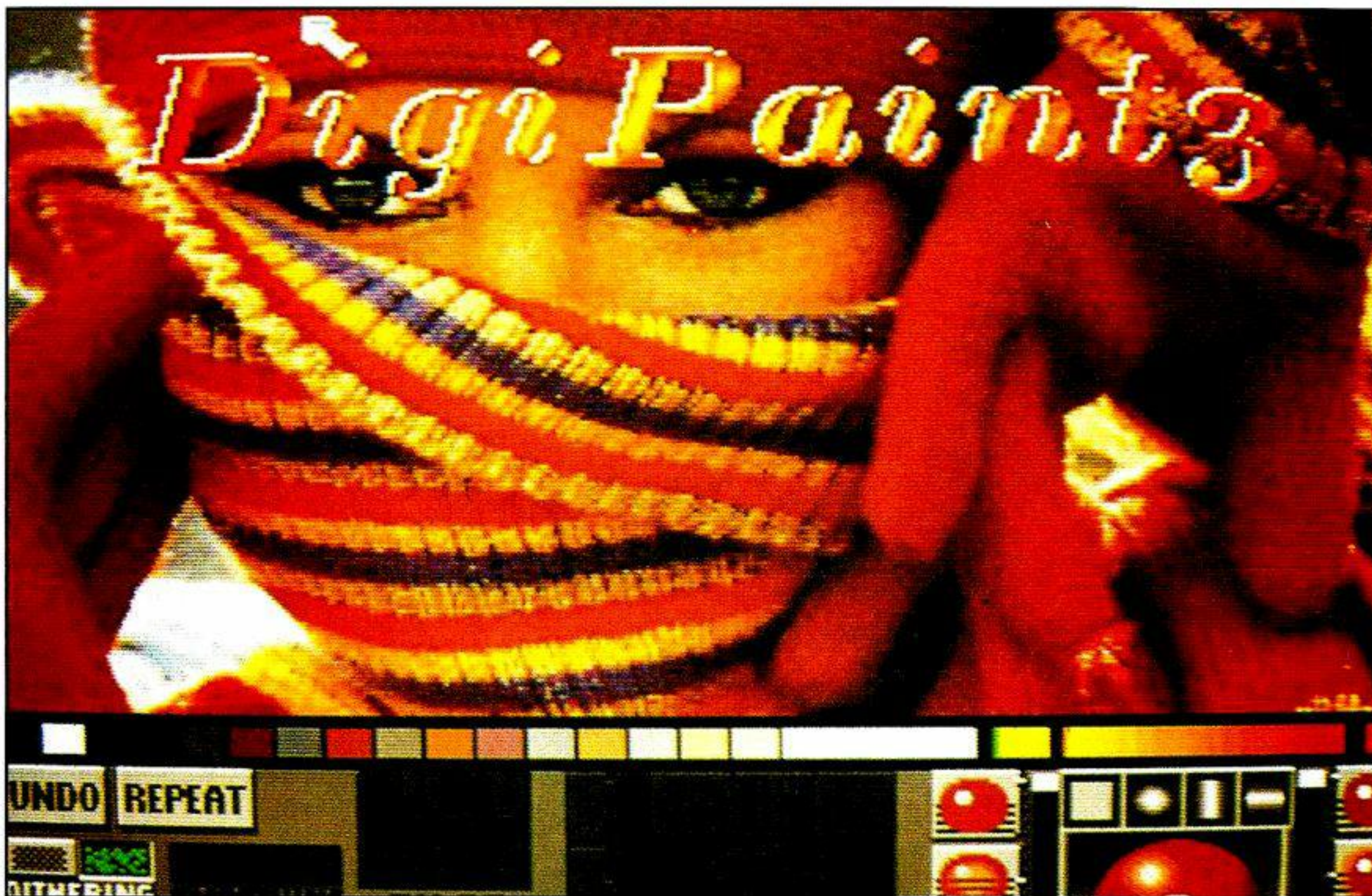
Lo si salvi su disco nella sua veste definitiva, e finalmente lo si lanci: la nostra pallina, i cui dati vengono prelevati dalle righe Data, entrerà subito in azione.

Il programma, dato il suo scopo puramente dimostrativo, è piuttosto semplice: basta far rimbalzare la palla tramite una "racchetta" il cui movimento è gestito dal mouse.

Ogni 5 "palleggi", la difficoltà viene incrementata diminuendo la distanza della racchetta dal bordo superiore. Se si fallisce il colpo, il minigioco termina mostrando quanti palleggi sono stati effettuati. Per evitarvi la digitazione di lunghe serie di Data, l'oggetto *Racchetta* è ottenuto in maniera molto grossolana ma, ora che disponete di un Data Maker, risulterà semplice disegnarsene una di tutto rispetto, e, con opportune migliorie, il programma potrà assumere una vera dignità di Game, anche se, come ovvio, con il solo Basic non si possono certo raggiungere i livelli di un *Arkanoid*.

Non si dimentichi, comunque, che sul numero 72 della nostra rivista è stato pubblicato un programma che trasforma un *Brush* di Deluxe Paint (e similari) in un oggetto gestibile da AmigaBasic. Inutile dire che anche i file prodotti da quel programma possono essere trasformati in *Data* dal Datamaker di queste pagine, con i risultati che si possono immaginare...

```
'                    L I S T A T O  1
'Trasforma un Bob o uno Sprite creati con l'Object Editor
'(o programmi similari) in un file Ascii di righe Data
'
'---- CARICA OGGETTO E PREPARA OUTPUT SU NUOVO FILE ----
WINDOW 2,"Bob & Sprite Datamaker",(30,70)-(580,115),18
label1:
CLS:PRINT:PRINT:INPUT"VOLUME/DIRECTORY DI INPUT";file1$
label2:
CLS:PRINT:PRINT:INPUT "NOME DEL FILE";file2$
IF file2$="" THEN GOTO label2
file1$=file1$+file2$
label3:
CLS:PRINT:PRINT:PRINT SPC(18)"DATI PER LINEA (5-15)";
INPUT dpl:IF dpl<5 OR dpl>15 THEN GOTO label3
CLS:PRINT:PRINT:INPUT"VOLUME/DIRECTORY DI OUTPUT";file3$
file3$=file3$+file2$+".data":CLS:PRINT"INPUT FILE ";
COLOR 3,2:PRINT SPC(4);file1$:COLOR 1,0
PRINT "DATI PER LINEA ";:COLOR 3,2:PRINT dpl:COLOR 1,0
PRINT"OUTPUT FILE ";:COLOR 3,2:PRINT SPC(3)file3$
COLOR 1,0:PRINT:PRINT SPC(24)"CORREZIONI (S/N)?";
correz:
a$=INKEY$:a$=UCASE$(a$)
IF a$<>"S" AND a$<>"N" THEN correz
IF a$="S" THEN CLEAR:GOTO label1
CLS:PRINT :PRINT :PRINT SPC(29)"ATTENDI..."
OPEN file1$ FOR INPUT AS 1:oggetto$=INPUT$(LOF(1),1)
CLOSE 1:totdati=LEN(oggetto$):OPEN file3$ FOR OUTPUT AS 1
PRINT# 1,"* NUMERO DATI = ";totdati
'------------ INVIA DATA DELL'OGGETTO AL FILE -----------
numlinee=INT(totdati/dpl):ultimo=dpl-1
FOR linee=1 TO numlinee*dpl STEP dpl:PRINT# 1,"DATA ";
 FOR dati=0 TO ultimo
  a=ASC(MID$(oggetto$,linee+dati,1)):GOSUB store
 NEXT dati
NEXT linee
'---------------- ULTIMA LINEA DATA -------------------
IF numlinee<>totdati/dpl THEN
 ultimo=totdati-numlinee*dpl:PRINT# 1,"DATA ";
 FOR dati=1 TO ultimo
a=ASC(MID$(oggetto$,(totdati-ultimo)+dati,1)):GOSUB store
 NEXT dati
END IF
'---------------- FINE OPERAZIONI ---------------------
CLOSE 1:CLS:PRINT :PRINT SPC(27)"TUTTO OK!":PRINT
PRINT SPC(10)"Aspetta che il drive si fermi ";
PRINT "e premi un tasto"
loop:
a$=INKEY$:IF a$="" THEN GOTO loop
WINDOW CLOSE 2:END
'---- SUBROUTINE MODIFICA ESTETICA E SCRITTURA DATI ----
store:
  a$=STR$(a)
  IF a<10 THEN PRINT# 1,CHR$(48);
  IF a<100 THEN PRINT# 1,CHR$(48);
  PRINT# 1, RIGHT$(a$, LEN(a$)-1);
  IF dati=ultimo THEN PRINT# 1,"" ELSE PRINT# 1,",";
  RETURN
```

```
;                LISTATO  2
' * NUMERO DATI =  106
DATA 000,000,000,000,000,000,000,000,000,000,000,002
DATA 000,000,000,021,000,000,000,010,000,024,000,003
DATA 000,000,000,000,000,000,000,000,000,000,003,248
DATA 000,000,015,254,000,000,031,255,000,000,015,254
DATA 000,000,003,248,000,000,000,000,000,000,000,000
DATA 000,000,000,000,000,000,003,254,000,000,015,255
DATA 128,000,063,255,224,000,127,255,240,000,127,255
DATA 240,000,127,255,240,000,063,255,224,000,015,255
```

```
DATA 128,000,003,254,000,000,000,000,000,000'
' DEMO DA LANCIARE SOLO DOPO AVER EFFETTUATO
' IL MERGE CON IL FILE "ball.data" CREATO
' CON IL DATAMAKER (listato 1)
'================================================
SCREEN 2,640,220,2,2
WINDOW 2,,(0,0)-(630,150),0,2
ON BREAK GOSUB fine:BREAK ON
'-------  LEGGE DATI CREATI COL DMAKER  --------
FOR x=1 TO 106:READ a:palla$=palla$+CHR$(a):NEXT
OBJECT.SHAPE 1,palla$:'assegna pallina a Bob 1
'----------  CREA BOB "RACCHETTA"  -------------
bar$=MKL$(0)+MKL$(0)+MKL$(2)+MKL$(24)+MKL$(4)
bar$=bar$+MKI$(24)+MKI$(3)+MKI$(0)
FOR x=1 TO 32:bar$=bar$+CHR$(255):NEXT:v=130
OBJECT.SHAPE 2,bar$:OBJECT.X 2,270:OBJECT.Y 2,v
'-----------------------------------------------
OBJECT.VX 1,80:OBJECT.VY 1,70:'velocita' palla
OBJECT.ON :OBJECT.START:'attiva gli oggetti
'----------  CONTROLLO COLLISIONI  -------------
detect:
m=MOUSE(0):OBJECT.X 2,MOUSE(1):id=COLLISION(0)
IF id=0 THEN detect
obj=COLLISION(id):vve=OBJECT.VY(1):vor=OBJECT.VX(1)
IF obj=-3 THEN PRINT "COLPO FALLITO!":GOTO fine
IF (obj=-2 AND vor<0) OR (obj=-4 AND vor>0) THEN
 OBJECT.VX 1,-vor
END IF
IF (obj=-1 AND vve<0) OR (obj=2  AND vve>0) THEN
 OBJECT.VY 1,-vve:count=count+1:tot=tot+1
 IF count=10 AND v> 30 THEN
  v=v-20:OBJECT.Y 2,v:count=0
 END IF
END IF
OBJECT.START:GOTO detect
'----  DISATTIVA TUTTO PRIMA DI CONCLUDERE  ----
fine:
PRINT :PRINT "PALLEGGI EFFETTUATI = ";INT(tot/2)
PRINT :PRINT :PRINT "premi un tasto"
loop:
a$=INKEY$:IF a$="" THEN loop
OBJECT.OFF:WINDOW CLOSE 2 :SCREEN CLOSE 2:END
```

DISEGNARE CON DIGIPAINT III

Non tutti gli utenti di Amiga sono in grado di usare i potenti strumenti offerti dai numerosi pacchetti grafici in circolazione; ecco le risposte ai dubbi più frequenti di chi disegna abitualmente con Digipaint 3

A cura di Luigi Callegari

COME SI CREA UNA SAGOMA DI TRACCIATURA

Per creare una *Custom Brush* si deve selezionare l'icona di strumenti *(Tools)* se non è gia visualizzata, poi si sceglie una qualunque delle sagome riportate *(triangolo, rettangolo, sfera...)* e subito dopo l'icona della *forbice*. Il cursore assume così la sagoma della forbice stessa e portandolo sullo schermo si può "ritagliare" una porzione di schermo facendola diventare la sagoma di tracciatura lasciata dal cursore premendo il pulsante sinistro. Per effettuare il ritaglio si deve delimitare l'area interessata premendo il pulsante sinistro del mouse e muovendolo mantenendolo pigiato, poi rilasciandolo si ritaglia la porzione.

EFFETTO TRASPARENZA

Digipaint III presenta,nelle schermate dimostrative sul disco, un simpatico effetto di "trasparenza". Per realizzarlo si clicki sul gadget *Controls:* si vedrà apparire il menu omonimo in basso sullo schermo (non si tratta di un menu a discesa di Intuition, di quelli che compaiono premendo il pulsante destro del mouse!).

All'estrema destra sono presenti i controlli di trasparenza, con uno slider verticale sulla sinistra, che serve per controllare il punto di trasparenza. Vediamo anche un grosso quadrato bianco con

quattro pulsanti in cima, che servono per indicare la direzione della luce rispetto alla faccia dell'ipotetica sfera rappresentata dal soggetto.

Infine, sulla destra, troviamo uno *slider* verticale che regola la trasparenza, come indicano anche le figure poste a lato. Si clicki sul secondo bottone da sinistra tra quelli posti al di sopra del quadrato bianco *(Point Hotspot)*. In questo modo compare un quadratino bianco sullo sfondo ombreggiato, rappresentante il punto di luce per il controllo della trasparenza, dove cioè inizia il centro della trasparenza.

Si trascini lo slider *Edge Transparency* in basso per rendere trasparente lo spigolo della sagoma, come si desidera. Infine si sposti il puntatore nel punto desiderato per lasciare la "brush" attuale in modo trasparente alla pressione del pulsante sinistro del mouse.

EFFETTI... SFEROIDALI

Uno degli effetti più stupefacenti offertoi dal pacchetto è certamente quello che permette di scrivere un testo qualunque su di una sfera, come è anche possibile vedere nel dimostrativo.

Digipaint III consente di usare il testo come una brush, quindi di illuminarla, ombreggiarla ed "appiccicarla" su figure solide.

Innanzitutto si deve produrre, il testo da rendere, come se fosse una sagoma di tracciatura entrando nel menu *Text* e clickando sull'apposito bottone di selezione. Quindi si clicka sul gadget *Stringa* e si immette il testo.

Se si desidera cambiare la fonte di caratteri, si deve clickare sul primo bottone, quello con la scritta *Fonts*, per accedere alle fonti memorizzate nella directory associata al device *Fonts:* (solitamente la directory *Fonts* del disco di sistema, ovvero quello usato per avviare il computer alla richiesta del Workbench).

Dopo avere terminato la digitazione, premendo *Return* o *Enter*, il testo viene reso secondo i colori ed il modo di tracciatura attuale *(Colorize, Range, Normal...)* selezionati dai menu *Intuition* collegati al menu. Per ottenere la tracciatura in *grassetto, corsivo* o *sottolineato*, bisogna clickare su di uno dei tre gadget con una **[A]** scritta nel corrispondente stile, posti accanto al gadget di stringa.

In seguito si deve scegliere dal menu (pressione del pulsante destro del mouse) *Brush* la opzione *Swap / Copy This Brush*.

Tale azione inserisce automaticamente la brush attuale, rappresentata dalla stringa di testo, in un buffer interno ed abilita la successiva opzione che dobbiamo usare.

Dal menu *Mode* si scelga *TxMap*, che sta per "texture mapping". In questo modo la stringa di testo viene usata come *mappa* per riempire le sagome.

DIGIPAINT III

Il pacchetto *Digipaint III* della *NewTek* è attualmente, insieme a *Photon Paint* II, il programma "di punta" per quanto riguarda la grafica *HAM*.Scritto interamente in linguaggio Assembler per garantire la massima velocità operativa, è in grado di lavorare sia in bassa risoluzione normale (320 x 256), sia in bassa risoluzione interlacciata (320 x 512).

In effetti si attende l'annunzio della commercializzazione anche in Italia della versione che supporta il cosiddetto modo *Dynamic Hires*, in grado di presentare 4096 colori in alta risoluzione (640 linee orizzontali) usando una particolare tecnica hardware e software, che cambia in pratica la *palette* di colori per ogni nuova linea di scansione.

Tale sistema è già supportato dal digitalizzatore hardware *Digiview III* della Newtek, che può generare direttamente immagini HAM in alta risoluzione partendo da telecamere in bianco e nero od a colori.

Digipaint III consente di avere un'area di tracciatura virtuale di 1024 x 1024 pixel, memoria *Chip* permettendo, con scorrimento continuo, sebbene con i nuovi modelli di Amiga dotati di 1 Megabyte di *Chip Ram* la massima dimensione di schermo sia effettivamente di 960 x 960 pixel. Vogliamo qui illustrare alcune semplici procedure di utilizzo delle entusiasmanti caratteristiche del pacchetto, che speriamo servano come soluzioni a problemi pratici dei suoi utenti, incapaci di leggere il manuale in lingua inglese o senza voglia di introdursi in strani meandri per comprendere anche le nozioni basilari.

Ricordiamo, innanzitutto, che Digipaint III lavora esclusivamente con *files grafici* in formato IFF e che per fare comparire il menu con gadget e requester vari si deve premere il tasto di *Help*.

Per avere un esempio si rientri nel quadrante *Tools*, si scelga *Fill* e poi la *sagoma rotonda*. Clickando in un punto dello schermo per tracciare la circonferenza (mantenendo premuto il tasto sinistro del mouse lo si sposti sino a ottenere le dimensioni volute, poi si rilasci il tasto sinistro) comparirà la parola *brush* come stampata sulla superficie della sfera.

Si noti che può essere necessario rimappare la maschera di colori della brush scegliendo *Palette / Remap* per fare comparire il risultato voluto.

TUTTI I COLORI DEL GRIGIO

Con *Digipaint III* è possibile ottenere la colorazione di una immagine in una scala di grigi.

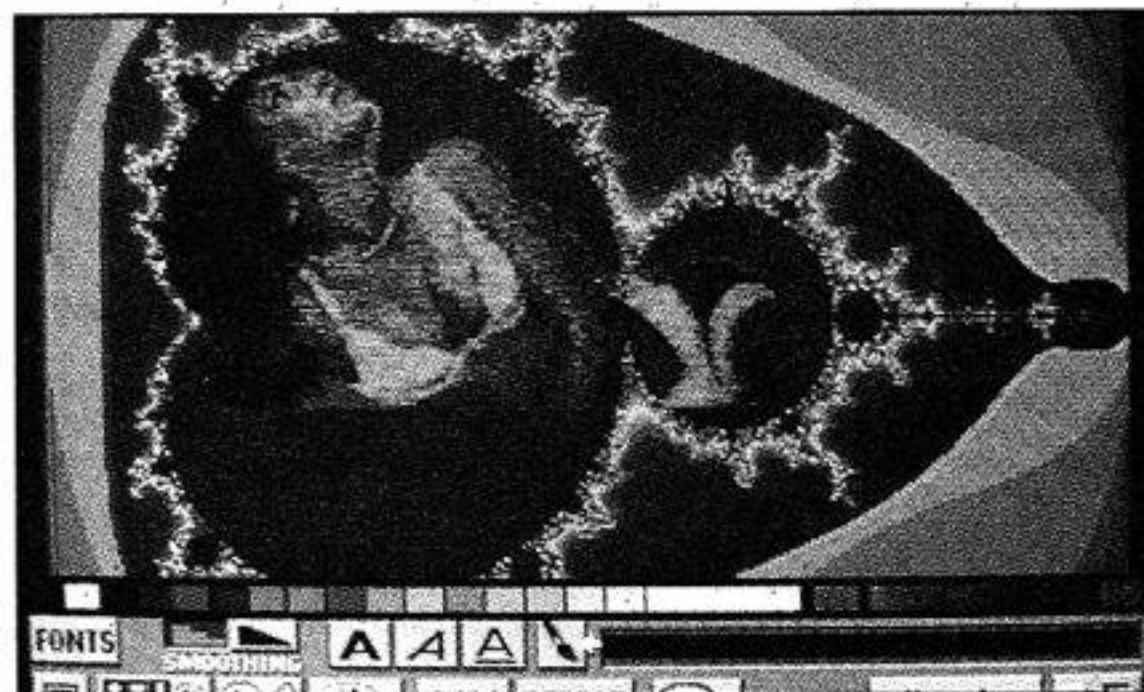
Si carichi una immagine a scala di grigi, come ad esempio il file *Lady* della directory *Images* del primo disco di Digipaint.

Si scelga quindi l'opzione *Colorize* dal menu *Mode* usando il pulsante destro del mouse. Si vada poi nel quadrante *Colors* (clickando sul bottone con il disegno di una *tavolozza* da pittore) e si agisca per ottenere (ad esempio) una gamma di sfumature rosa: si porti il puntatore del mouse al di sopra di uno dei colori rosa nei quadrati con le sfumature di colori posti sulla destra dello schermo.

Quindi si clicki il pulsante sinistro e lo si mantenga premuto muovendo il puntatore sulle sfumature.

Una volta scelto il colore appropriato si ritorni al quadrante di *Tools*, si clicki su *Fill* e, poi, sulla sagoma di tracciatura continua a mano libera (la prima a sinistra quando il *Fill* è stato selezionato e compare verde su nero). Delimitando la faccia della *lady*, e muovendo il mouse col pulsante sinistro premuto, si riempie di colore il disegno in modo concorde con la scala di grigi originale.

SELEZIONE CROMATICA

Per leggere un colore dallo schermo, allo scopo di usarlo come colore corrente, si deve attivare il quadrante della *tavolozza* cromatica, poi clickare sul bottone nella linea inferiore riportante *Pick*.

Portando il mouse su di un qualunque punto dello schermo, anche su di una porzione ingrandita con l'opzione di ingrandimento (bottone con *lente* da filatelico) e clickando col pulsante sinistro per un istante, il colore viene letto ed usato come colore attuale.

Viene infatti presentato anche sopra i tre slider cromatici verticali RGB nel quadrante.

UNIONI

L'opzione *RubThru* del menu *Mode* serve, essenzialmente, a unire creativamente due immagini separate in *modo selettivo*.

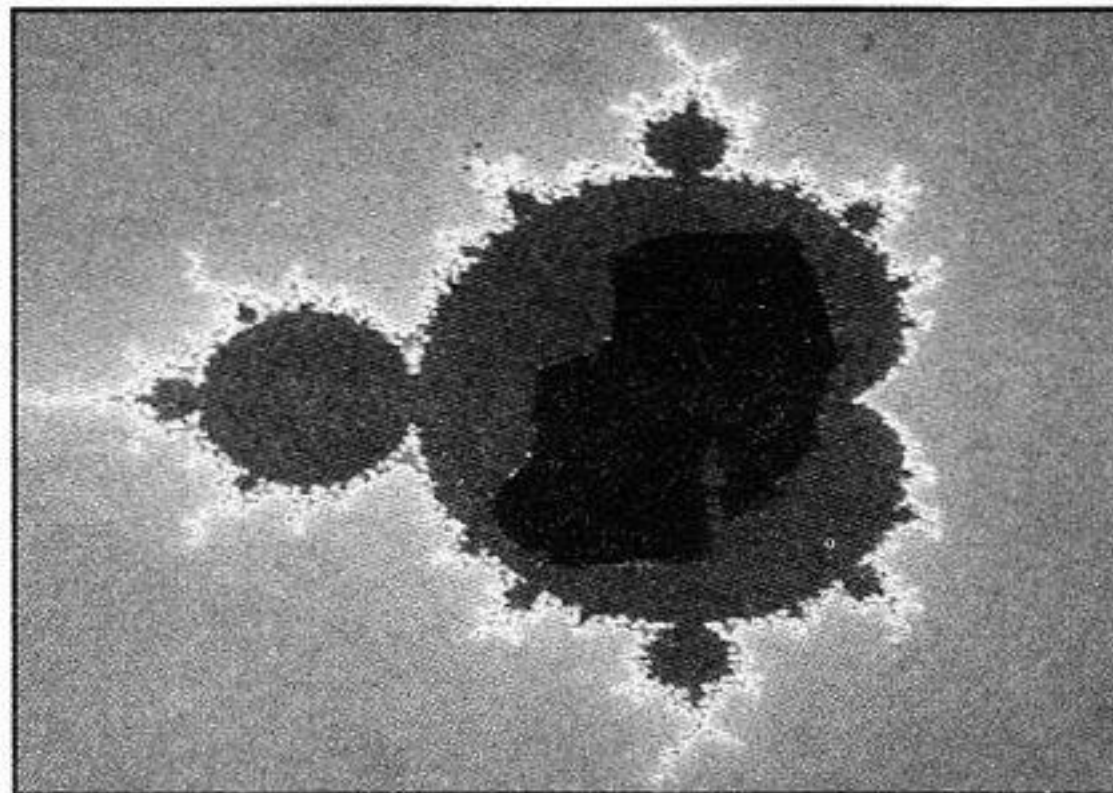
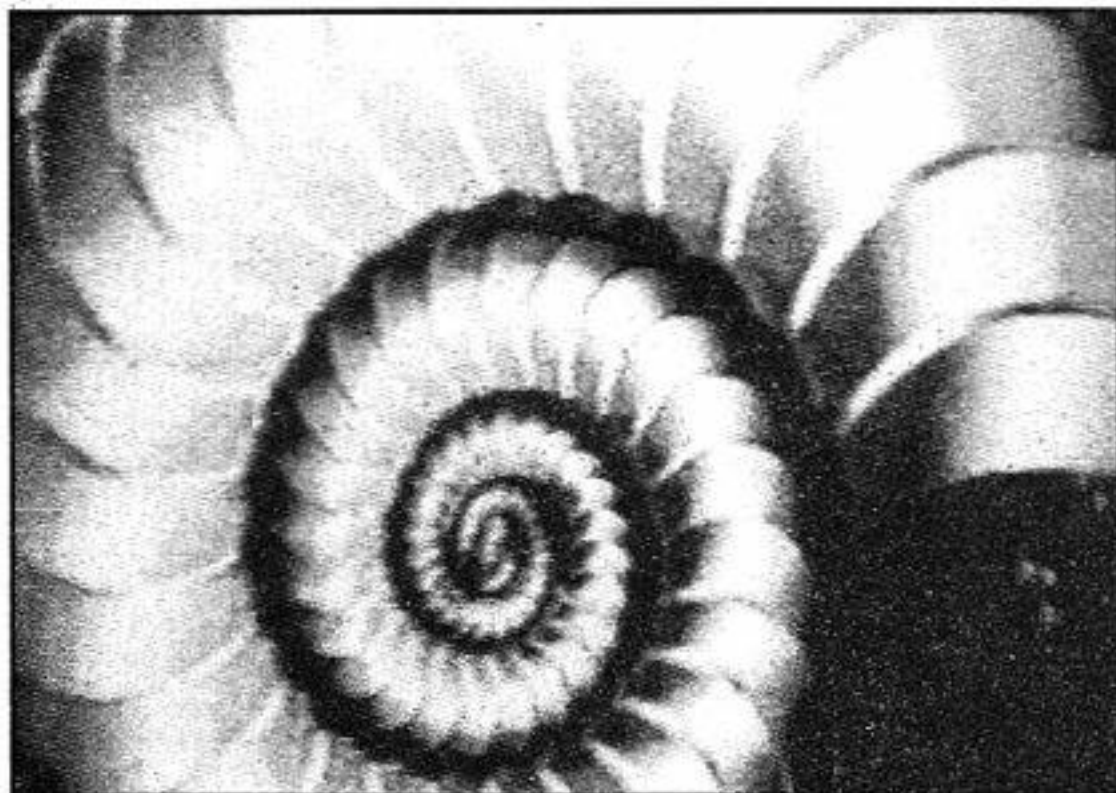
Si carichi, ad esempio, l'immagine *Fashion* dalla directory *Images* dal disco di Digipaint III, poi si scelga *Copy This Picture* dal menu *Picture / Swap* per ricopiare la schermata nel buffer interno.

Si scelga il colore bianco dalla tavolozza a *sedici* tinte (quello orizzontale sopra i bottoni, sempre visualizzato) e si scega *Colorize* dal menu *Mode:* premendo il tasto **[w]** si renderà tutto lo schermo in bianco e nero.

Poi si clicki sul gadget *Controls* per lavorare sui controlli di trasparenza.

Si attivi il *punto di fuoco* puntiforme (secondo gadget sopra il quadrato bianco) per fare comparire la sfera rossa col quadratino bianco, poi si sposti lo slider *Edge Transparency* (l'ultimo slider verticale a destra) nel punto inferiore della sua corsa.

Si scelga poi *RubThru* dal menu Mode.

In questo modo, premendo ancora [w], si vedrà ricolorare tutto lo schermo, usando però lo schermo sottostante come riferimento ed ottenendo così un effetto creativo in base al modo selezionato attualmente (si provi ad esempio a selezionare *Range* prima di dare il "w"). Con questa tecnica si possono ottenere dei "ritagli" o dei fotogrammi usando *Undo* o RubThru sullo schermo sottostante.

I DISEGNI DI FRA' MARTINO

La procedura di Algomartin, implementata in Amigabasic, è facile da digitare ed in grado di generare decine di fantastiche figure astratte

Salvatore Recupero

Il programma si basa sull'algoritmo del matematico inglese *Martin;* esso riproduce, attraverso la visualizzazione di punti, figure molto belle a vedersi, con effetti grafici imprevedibili. Ma esaminiamo il programma che all'inizio, dopo la presentazione, vi chiederà di inserire alcuni valori:

Se volete il grafico a colori (scelti, a caso tra otto, mediante la funzione Rnd) oppure con un solo colore (il bianco) dovrete digitare, rispettivamente, 0 oppure 7.

Subito dopo sarà la volta dei sei parametri *A, B, C, I, AX, AY* che ora descriviamo in dettaglio.

A, B, C: sono le variabili che ogni volta determineranno la figura. Si consiglia di introdurre valori a caso, preferendo numeri piccoli; oppure servirsi dei parametri consigliati dallo stesso programma.

I: è parametro che serve a definire la grandezza totale della figura sullo schermo; è in relazione con i tre parametri visti prima e può capitare che un ingrandimento eccessivo porti la figura fuori schermo; al contrario, un ingrandimento modesto può impedire di far vedere al meglio il disegno ottenuto.

Comunque, con un po' di esperienza, arriverete a scegliere l'ingrandimento ideale. La gamma dei valori può variare da 1 all'infinito (o quasi...). Ovviamente più è grande il valore digitato, più grande sarà la figura sullo schermo.

AX, AY: sono le coordinate sulle quali sarà stampata la figura. AX si riferisce all'asse orizzontale, AY a quello verticale. Fate attenzione a non esagerare con le coordinate, altrimenti la figura generata potrebbe andare fuori schermo o non comparire affatto. E' chiaro che con certi dati bisognerà effettuare dei tentativi per trovare le coordinate migliori; vi consigliamo, in linea di massima, di adottare le coordinate al centro dello schermo 280 - 285 per l'asse X, 80 - 90 per l'asse Y.

Non appena avrete inserito tutti i dati (separati uno dall'altro con la virgola) e dopo aver premuto Return, il programma inizierà a visualizzare la figura.

Durante il procedimento sarà attivo il menu *Tools* con il quale potrete disegnare una nuova figura (cambiando i parametri) oppure terminare il programma; l'algoritmo, infatti, procede all'infinito e non è prevista una fine.

La formula su cui è basata la procedura si trova alla linea *Formula:* mentre la linea *Stampa:* si incarica di disegnare i singoli punti.

Ovviamente l'algoritmo, per la sua semplicità, può essere variato in molti modi, secondo la vostra fantasia ed esperienza.

A parte la sezione del colore e della risoluzione, modificabile facilmente cambiando i valori di *Screen*, sono possibili altre modifiche come per esempio la funzione matematica della formula, in partenza *Sqr*, che può essere convertita in *Sin, Cos* ed altre; oppure l'operazione contenuta nella variabile *Y1* (cioè A - X) sempre alla linea *Formula:* potrebbe essere cambiata in B - X.

Si possono anche disegnare quadrati invece di punti (basta tramutare le *Pset* del programma con *LINE*), oppure circonferenze con il comando *Circle*.

E' bene precisare che occorrono alcuni minuti prima che le figure sullo schermo comincino ad assumere aspetti veramente interessanti; ma avendo a disposizione un compilatore basic (o eventualmente implementandolo in C) i risultati saranno eccellenti.

```
REM AlgoMartin 2.0 15/8/89 by Rec Sal

crzmn: REM crea il menu tools
MENU 5, 0, 1, "Tools"
MENU 5, 1, 1, "Nuovi dati"
MENU 5, 2, 1, "Fine"
ON MENU GOSUB scelta
MENU ON

parametri:  REM raccolta di alcuni dati
CLS
COLOR 3, 0: CALL centro (2, "AlgoMartin") : suono: LINE (252, 16) - (338, 16)
LINE (250, 100) - (600, 180) , 3, b
LOCATE 14, 44: COLOR 1, 0: PRINT "Parametri consigliati"
LOCATE 16, 33: COLOR 3, 0: PRINT "1)  5, 0.8, 1.6, 10, 200, 80"
LOCATE 17, 33: PRINT "2)  0.3, -3, 2.7, 10, 220, 80"
LOCATE 18, 33: PRINT "3)  -0.7, -1, 4, 10, 250, 80"
LOCATE 19, 33: PRINT "4)  8, -4, -0.48, 10, 250, 80"
LOCATE 20, 33: PRINT "5)  1.1112, -2.2, 0.6, 15, 285, 85"
LOCATE 21, 33: PRINT "6)  3.3331, -1.1, 0.3, 14, 285, 85"
LOCATE 22, 33: PRINT "7)  4, -1, -0.7, 13, 285, 85"

main: REM inizializ. dati e programma
x = 0: y = 0
LOCATE 4, 1: COLOR 1, 0: INPUT"Grafico ad 1 colore od a 8  (0/7) ";col%
IF col% <> 0 AND col% <> 7 GOTO main
FOR scr = 1 TO 57
  SCROLL (0, 24) - (280, 32) , -5, 0
 NEXT scr
LOCATE 4, 8: INPUT"A, B, C, I, AX, AY ";a, b, c, i, ax, ay
FOR scr = 1 TO 90
  SCROLL (0, 24) - (480, 32) , 5, 0
 NEXT scr
 scrolling (-15)
CLS: SCREEN 1, 640, 256, 3, 2
WINDOW 2, "AlgoMartin", , 0, 1

formula: REM formula algoritmo
x1 = y-SGN (x) *SQR (ABS (b*x-c) )
y1 = a-x: c1 = 1 + RND*col%

stampa: REM stampa ogni punto grafico
PSET (x*i + ax, y*i + ay) , INT (c1)
PSET (x1*i + ax, y1*i + ay) , INT (c1)
x = x1: y = y1
GOTO formula

fine:
END

REM procedure
SUB centro (r%, mes$)  STATIC
col% = 76/1
LOCATE r%,  (col%-LEN (mes$) ) /2: PRINT mes$
END SUB: REM stampa al centro una stringa

SUB suono STATIC
SOUND 292, 2, 100, 1: SOUND 523.28, 2, 100, 2: SOUND 659.28, 2, 100, 3
END SUB: REM emette un suono

SUB scrolling (ab%)  STATIC
 FOR i = 1 TO 15
     SCROLL (0, 1) - (615, 200) , 0, ab%
    NEXT i
END SUB: REM effettua lo SCROLL dello schermo
SUB pausa (p%)  STATIC
FOR i = 1 TO p%: NEXT i
END SUB: REM crea una pausa secondo il valore inserito

scelta: REM scelta da menu
IF MENU (1)  = 1 THEN WINDOW CLOSE 2: SCREEN CLOSE 1: RETURN parametri
IF MENU (1)  = 2 THEN WINDOW CLOSE 2: SCREEN CLOSE 1: RETURN fine
```

I COMPRESSORI DI AMIGA

Operando con i modem, o volendo risparmiar dischetti, capita di dover realizzare la "quadratura del cerchio", cioè ridurre il numero di byte da elaborare o archiviare

di Luigi Callegari

I cosiddetti *compressori* sono programmi in grado di ridurre le dimensioni occupate da *files di dati*, siano essi files Ascii, programmi vari (scritti in qualsiasi linguaggio) od interi dischetti, allo scopo di far occupare lo spazio minore possibile, pur mantenendo intatti i dati stessi.

I complementari *decompressori*, che quasi sempre sono incorporati nello stesso programma compressore, effettuano l'operazione opposta, riportando i dati del file o del disco al formato originale.

Come qualcuno avrà intuito, questo tipo di programmi può essere usato vantaggiosamente, in particolare, quando si devono inviare dati via Modem, in quanto consentono di ridurre il tempo del collegamento a beneficio delle proprie tasche.

Dal momento che all'estero le BBS (cioè le banche dati telefoniche totalmente automatizzate) sono sviluppate da molto tempo, esistono parecchi programmi in circolazione, studiati appositamente per svolgere il meglio possibile questo compito. Caratteristica comune di (quasi) tutti i compressori è l'essere di pubblico dominio, infatti sono rintracciabili facilmente sia nelle BBS, sia nelle varie collezioni disponibili gratuitamente (**Fish Disk**, ad esempio).

I programmi di compressione più diffusi, tra quelli studiati appositamente per gli utenti di modem possessori di Amiga, sono **Arc** e **Zoo**. Il primo è forse il più vecchio e glorioso programma del suo genere, esistente in versioni apposite per (praticamente) qualunque computer e sistema operativo (MS/DOS, AmigaDOS, UNIX, Macintosh...), mentre il secondo rappresenta una evoluzione di Arc, col quale è compatibile a livello di comandi ("verso il basso"), pur essendo più efficiente e con un maggior numero di opzioni di lavoro.

Attualmente un nostro connazionale, *Paolo Zibetti*, ha realizzato una nuova versione di compressore inserita nel pubblico dominio internazionale (Fish Disk *293*) che ha riscosso un immediato successo grazie alle sue eccezionali prestazioni e caratteristiche, chiamato **LHarc**. Anch'esso è compatibile verso il basso con *Arc*, pur possedendo un algoritmo di compressione molto più efficiente e veloce.

Altro programma usato frequentemente nelle BBS internazionali è il famoso **Warp**, che funziona trasformando i dati contenuti in un disco intero, o entro un dato numero di settori, in un file binario tranquillamente inviabile via Modem. Ciò consente di inviare (o archiviare) dischi in formati non standard (anche protetti, raramente), che possono poi essere riscritti usando Warp stesso oppure il programma separato **Unwarp**, studiato appositamente per funzionare più velocemente. Ovviamente anche Warp esegue una compressione dei dati letti.

Infine citiamo **Power Packer** di *Nico Francois*, software di pubblico dominio rintracciabile anche in *Amigazzetta*, che è in grado di comprimere files eseguibili

COMANDI DI ZOO SUPPLEMENTARI A QUELLI DI ARC

FORMA 1	DESCRIZIONE	FORMA 2
-add	Aggiunge files in archivio	aP:
-extract	Estrae files dall'archivio	x
-move	Sposta files dall'archivio	aMP:
-test	Verifica integrità archivio	xNd
-print	Estrae files dall'output	xp
-delete	Cancella files dall'archivio	DP
-list	Elenca i contenuti degli archivi	VC
-update	Aggiunge files nuovi	aunP:
-freshen	Aggiorna con nuovi files	auP:
-comment	Aggiunge commenti ai files	c

I COMANDI DI ARC

a = Aggiunge files all'archivio
b = Conserva una copia di sicurezza dell'archivio
c = Converte un elemento nel nuovo metodo di packing
d = Cancella files dall'archivio
e = Estrae files dall'archivio
l = Elenca i files in archivio
m = Sposta files all'archivio
n = Sopprime le note ed i commenti
p = Copia files dall'archivio a stdout
r = Esegue files nell'archivio
t = Verifica integrità dell'archivio
u = Aggiunge files all'archivio
v = Elenca verbosamente i files in archivio
w = Sopprime i messaggi di avvertimento
x = Estrae files dall'archivio

N.B: alcuni comandi sono ottenibili con due lettere diverse

e di dati con una interfaccia utente molto sofisticata. Non è particolarmente utile a chi deve inviare dati via modem, in quanto non può raggruppare in un solo file vari files come Arc, Zoo e LHarc, ma è indispensabile a chi deve ridurre le dimensioni di files eseguibili. Inoltre è un programma sofisticatissimo dal punto di vista dell'interfaccia utente, consentendo addirittura l'elaborazione differita (*batch*) dei files da comprimere (lavora anche in assenza dell'operatore, memorizzando sequenze di eventi mouse/tastiera e ripetendoli da sè).

E' veramente utile anche per decomprimere files trattati con programmi di *crunching* diffusi tra gli hacker europei.

COME FUNZIONANO

Arc, Zoo e LHarc sono programmi che leggono un file per volta, lo comprimono e lo inseriscono in un file detto *di archiviazione*, solitamente col nome terminato da un suffisso (*.arc*, *.zoo* e *.lzh* rispettivamente). Questo file conserva sia i *dati* compressi veri e propri, sia *informazioni* estese su di essi (nome, data di inserimento, tecnica di compressione usata ed altro), rileggibili tramite apposito comando *(l)*. Non si tratta ovviamente di un file eseguibile, ma di una sequenza di bytes comprensibili soltanto ai programmi di compressione che li ha generati.

I programmi sono sviluppati in versioni per vari sistemi operativi. Ciò significa che se qualcuno usa, ad esempio, Zoo sul proprio Macintosh o IBM per inserire in una BBS una serie di files, un altro utente può usare un Amiga od un Apple per riportare tali files alle dimensioni originali senza problemi di alcun tipo, una volta letto via modem tale file.

Si badi che, in nome di questa uniformità, vi sono delle caratteristiche di funzionamento particolari, che potrebbero risultare inusitate a chi è ormai abituato ad usare AmigaDOS.

Ad esempio, i ben noti caratteri *jolly*, cioè cancelletto (#), punto di domanda (?), ed asterisco (*), sono presi in considerazione anche da Ms/Dos e da Unix. Inoltre i nomi dei files, spesso, vengono tagliati ai primi dieci o quindici caratteri e se ne considera importante il suffisso.

Ad esempio, per default *Arc* assume che il file di archiviazione sul quale lavorare termini nel nominativo con la stringa *".arc"*.

Ricordiamo che il *suffisso* di nome di un file è tipicamente una sequenza di tre caratteri che segue il punto (.), eredità di Ms/dos principalmente.

Un'ultima precisazione riguarda il fatto che i compressori memorizzano un file per nome, quindi non è lecito memorizzare due volte un file con lo stesso nome.

Ad esempio, se in due directory separate abbiamo due files differenti chiamati ambedue *Cattorelli*, la prima volta il programma lo archivierà normalmente, ma al momento della compressione dei files nella seconda directory segnalerà *File already present* e non lo inserirà. Ciò potrebbe comportare la perdita di preziosi files, quindi si faccia attenzione e non

si comprimano troppi files disparati in uno stesso file se non si è ben sicuri che siano dotati tutti di nomi differenti.

ARC

Il capostipite dei compressori d'archivi è stato diffuso originariamente sul **Fish Disk 70** in versione *0.23*, completamente compatibile con la famosa versione 5.00 di MS/DOS.

Ricordando che digitando il suo nome senza parametri da *Shell*, e battendo Return, si ottiene l'elenco dei comandi disponibili, la sua sintassi di chiamata è:

ARC <comando> <nomearc> [<argomento>]

in cui con *<comando>* indichiamo una lettera che esprime il comando da eseguire (vedi Figura 1), maiuscola o minuscola. *Con <arc>* si indica il nome del file di archiviazione, anche senza estensioni (caratteri dopo il punto nel nome), nel quale caso si assume per default "*.arc*".

Ad esempio, con la linea...

*Arc a Ormellese *.**

...si *aggiungono* al file di archivio *Ormellese.arc* tutti i files presenti nella directory corrente. I simboli di asterisco indicano "qualunque stringa", secondo le convenzioni dei "caratteri jolly" di MS/DOS. Analogamente il comando...

*ARC m RAM: Gente df1: pil/ *.c*

...muove nel file di archivio *Gente.ARC* presente nel *Ram Disk* tutti i files della directory *pil* presente nel dischetto inserito nel primo drive esterno (o nel secondo interno di un A2000) che abbiano il nome terminante col suffisso *".c"*. Con "muovere" o "spostare", termini rubati all'inglese nel gergo informatico italiano, si intende che i files accettabili (in questo caso quelli con i nomi terminanti per *.c*) vengono compressi, inseriti nel file di archivio e cancellati.

Invece con "estrarre un file dall'archivio", ovvero col comando "*x*", si intende il leggerlo dal file di archiviazione dove è stato compresso e memorizzato, e riportarlo al formato originale. Ad esempio...

*ARC x df0: testi *.doc*

...scandisce il file di archiviazione *Testi.arc*, assunto presente nel disco inserito nel drive interno, ricavandone tutti i files con nomi terminanti in *".doc"*. Si noti che ciò presuppone implicitamente che tutti i files vengono memorizzati nell'archivio col nome originale, come difatti accade.

Per leggere i nomi e le statistiche di tutti i files inseriti in un archivio, si usa qualcosa come...

ARC l archi
ARC v sys: mede

Il primo genera un elenco normale dei files compressi nell'archivio *"archi.ARC"* nella directory corrente; il secondo un elenco molto crcostanziato dei files nell'archivio *"mede.ARC"*. In quest'ultimo vengono elencati ordinatamente: *nome* del file, *lunghezza* in bytes originale, *tipo* di compressione usata, *riduzione* percentuale, *dimensione* in bytes nell'archivio, *data* di inserimento, *ora* di inserimento, *codice* di checksum.

La terza colonna riporterà *Squeezed* oppure *Crunched*: Il primo indica che il file è stato compresso con l'algoritmo di *Huffman*, il secondo con quello di *Lempel-Zev*. La scelta del tipo di compressione viene effettuata autonomamente da ARC al momento dell'inserimento del file in archivio, in modo da ottenere la massima riduzione possibile; in genere Lempel Zev funziona meglio con i files corti.

ZOO

Zoo è una evoluzione di Arc scritta per Amiga da *Rahul Desi* e rintracciabile, oltre che in numerose BBS, anche sul Fish Disk numero *164*. E' scritto in C ed

GLI SWITCHES DI LHARC:

- -p = Pausa dopo il caricamento
- -m = Nessun messaggio di richiesta
- -x = Usa nomi di file con estensione
- -n = Disabilita messaggi di lavoro
- -w = Indica la directory di lavoro
- -P = Fissa la priorità
- -a = Considera gli attributi di file
- -u = Converte nomi di file in maiuscolo
- -r = Raggruppa files ricorsivamente

esistono da tempo versioni per Apple, MS/DOS, UNIX ed altri sistemi operativi.

Zoo può memorizzare ed estrarre selettivamente files dagli archivi come Arc, ma usa un algoritmo di Lempel-Ziv modificato, che consente una riduzione dal 20% all'80% del file sorgente. I dati possono essere recuperati anche da archivi danneggiati e si può sottoporre a compressione il file di archivio per recuperare lo spazio liberato da files archiviati eliminati.

Essendo compatibile con Arc, tutti gli esempi ed i comandi visti per quest'ultimo valgono anche per Zoo. Il formato generico di Zoo risulta decisamente complesso da imparare a memoria:

Zoo {acfDeghlLPTuUvVx} [aAcCdEfglmMnNoOpPqSu1:/.@+-=] archivio [file]

Zoo -comando archivio [file]...

Digitando *Zoo h* [+ return] si ottiene un elenco completo delle funzioni disponibili. I caratteri racchiusi tra parentesi graffe sono *comandi*, quelli racchiusi tra parentesi quadre sono *modificatori* dei comandi. Ovviamente non tutti i modificatori funzionano con tutti i comandi.

I BENCHMARKS

Abbiamo provato a confrontare tra di loro i programmi di compressione files più diffusi, ottenendo dei significativi "benchmarks", dai quali ognuno può trarre le proprie conclusioni.

Si noti che i programmi analizzati sono comunque difficili da paragonare, in quanto diversi per concezione tra di loro: *LHarc, Zoo e Arc* sono dei compressori di archivi veri e propri, mentre *Power Packer* è in grado di trattare agevolmente anche files eseguibili con un notevole grado di sofisticazione ed una interfaccia *Intuition*izzata molto evoluta. Inoltre PowerPacker genera un file eseguibile con codice di autodecompressione incorporato da un file eseguibile, mentre gli altri programmi generano un file archiviato con dei dati supplementari che consentono di agganciarvi altri files compressi.

I benchmarks sono stati eseguiti sui files dei programmi DiskMaster V1.3 di 61860 Kbytes e l'editor/assembler ArgEd della Argonaut di 149600 bytes, contenenti ambedue codici assembly, testi ASCII e dati grafici. Per rilevare i tempi è stato utilizzato il programma RTimer del pacchetto Disk Mechanic della Lake Forest Logic.

I computer ospiti erano un Amiga 2000 con scheda GVP Impact A3000 con processore 68030 a 25 MHz per il primo benchmark ed un normale Amiga 500 per il secondo.

I tempi riportati sono in secondi. Si noti che il programma Power Packer V2.3b di Nico Francois è stato usato due volte con efficienze di lavoro diverse ("best" e "good").

Programma	Tempo	Bytes	Riduzione
PPacker V2.3b (best)	126.5	33448	46%
PPacker V2.3b (good)	41.8	34144	45%
LHarc V1.00	17.9	32002	48%
Zoo V2.00	3.1	40969	34%
Arc V5.00	10.5	45847	26%

Programma	Tempo	Bytes	Riduzione
PPacker V2.3b (best)	1707	70904	52%
PPacker V2.3b (good)	567	72944	51%
LHarc V1.00	429	67006	55%
Zoo V2.00	34	90986	39%
Arc V5.00	162	108524	27%

Con *"-comando"* si intende non una singola lettera ma una stringa tra quelle riportate in Figura 2, adatte all'uso da parte dei non esperti, che quindi vengono chiamati *Novice Command,* in contrapposizione agli *Expert Commands,* che sono in pratica le stesse funzioni, ottenute però con singole lettere (come Arc).

LHarc

La versione di LHarc per Amiga in circolazione attualmente è la *1.00,* compatibile con la *1.13* per MS/DOS, ed è stata scritta da *Paolo Zibetti* (FidoNet 2: 331/ 101.6).

Anche LHarc è un programma di archiviazione di files come Arc e Zoo, con i quali è compatibile in tutti i comandi fondamentali. Il suo punto debole è la velocità: *Zoo V2.0,* ad esempio, è più veloce. Comunque, se si preferisce l'efficienza di compressione, è il programma migliore in circolazione per Amiga (e la decompressione è più veloce della compressione).

La sua sintassi, da *Shell,* è come segue:

LHarc [Switches] <Comando> <Archivio> [Destinazione] [Maschera files]

...in cui le parentesi quadre delimitano parametri opzionali.

Con *Comando* si indica una lettera secondo quanto visto per Arc (figura 1), con identica funzione, tra le seguenti:
e, x, l, v, p, t, a, m, d, u, f.

Con *Archivio* si indica il nome standard AmigaDOS (con eventuale path) del file di archiviazione, al quale viene, per default, aggiunto il suffisso *".LZH"* se non diversamente specificato.

Analogamente con *Destinazione* si indica la directory di destinazione alla quale vengono inviati i files estratti con i comandi **x** oppure **e**.

Con *Maschera files* si indica una stringa che esprima i nomi di files interessati all'operazione.

I caratteri jolly sono "#", "?" e "*" che è sinonimo di "#?" (qualunque numero di qualunque carattere). Gli *switches,* cioè i modificatori di funzionamento, propri di LHarc, sono invece riportati nel terzo riquadro.

Warp

Il programma Warp della SDS, giunto alla versione *1.11* (rintracciabile sul *Fish Disk 243*) è quasi banale da usare.

Le sue sintassi di chiamata da Shell sono:

Warp read <Start> <end> <nomefile>

Warp write <nomefile>

Con la prima linea si impartisce di leggere *(Read)* i dati contenuti nel disco inserito nel drive interno, dal settore *<Start>* al settore *<End>* opzionalmente, scrivendo il risultato nel file *nomefile*.

Ad esempio, con le linee:
Warp Read 10 20 Df0: Piera
Warp Read Df1: Filosofia
Warp Write Df1:Casorate
si ottiene, rispettivamente:

(1) la scrittura nel file *Piera* del disco inserito nel drive interno di Amiga dei contenuti dei settori tra il decimo ed il ventesimo compresi.

(2) la scrittura dell'intero contenuto del disco inserito nel drive interno nel file *Filosofia* sul disco inserito nel primo drive esterno (o secondo interno di A2000)-

(3) di riscrivere il disco inserito nel drive interno leggendo dal disco DF1 il file *Casorate*, che ovviamente deve essere stato generato da *Warp*.

Si noti che Warp lavora sempre sul disco inserito in DF0: e che non ha molto senso, normalmente, specificare i settori di inizio e di file della lettura, in quanto un disco AmigaDOS parzialmente salvato serve a ben poco, quindi bisogna sempre salvare da 0 a 79 (default).

Insieme a Warp si trova spesso anche *UnWarp*, che è più veloce nell'effettuare l'operazione di *Warp Write* di un file, e verifica, nel contempo, l'eventuale presenza di quindici tipi di **virus**.

Si ricordi che i virus di bootblock di Amiga giacciono nei settori 0 e 1 del disco, quindi *Warp* potrebbe tranquillamente archiviarli e ricostruirli al momento del Warp Write!

Con UnWarp si è parzialmente protetti da questo rischio, anche se alcuni nuovi *Link Virus*, quelli che non si attivano al boot ma si agganciano ai files eseguibili, non sono ancora stati previsti e quindi non possono esser riconosciuti.

NOTE STORICHE E TECNICHE

ARC è stato scritto per la prima volta nel 1985 dalla System Enhancement Associates (21 New Stree, Wayne, NJ 07470, USA) e da allora è stato sviluppato probabilmente per qualunque tipo di computer in grado di collegarsi ad una rete. Sono poi state introdotte versioni sempre nuove di Arc, ma le sue caratteristiche nel mondo Amiga sono superate da quelle di altri programmi di più recente creazione (Zoo, LHarc), che fanno uso di algoritmi di compressione nuovi. Non dimentichiamo che l'informatica e le branche della matematica dedite alla formulazione di algoritmi iterativi ed adatti ai computers sono di nascita relativamente recente, quindi nascono invenzioni ed idee nuove piuttosto frequentemente!

Comunque, alla base della nascita di ARC e dei compressori di archivi vi sono un numero ristretto di personaggi e di concetti, dai quali è nata la filosofia e la tecnologia di base di questo tipo di software. Citiamo Brian Kernighan (sì, quello del linguaggio C!) e Paul Plauger che hanno formulato le prime idee nel libro "Software Tools", Dick Greenlaw che ha scritto i programmi di pubblico dominio SQ e USQ (già pubblicati da Amigazzetta), Kent Williams che ha trasformato delle formule matematiche di Lempel-Zev in un programma funzionante per la prima volta e David Schaderer che ha formulato e pubblicato su PC Tech Journal il primo algoritmo di CRC (Cycle Redundancy Check) adatto a piccoli sistemi computerizzati, che ha dato la prima idea funzionante per aumentare l'affidabilità dei sistemi di compressione di archivi.

QUANTO COSTA IL TUO COMMODORE

Amiga 2000 - L. 2.715.000

Microprocessore Motorola MC68000 - Clock 7.16MHz - Kickstart ROM - Memoria RAM: 1 MByte - 3 chip custom per DMA, Video, Audio, I/O - 5 Slot di Espansione Amiga Bus 100 pin AutoconfigTM - 1 Slot di Espansione 86 pin per Schede Coprocessore - 2 Slot di Espansione compatibili AT/XT - 2 Slot di Espansione compatibili XT - 2 Slot di Espansione Video - 1 Floppy Disk Drive da 3 1/2", 880 KBytes - Porta seriale RS232C - Sistema Operativo single-user, multitasking AmigaDOS - Compatibilità MS-DOS XT/AT disponibile con schede interne Janus (A2088 - A2286) - Monitor escluso

Amiga 500 - L. 995.000

Microprocessore Motorola MC68000 - Clock 7.16 MHz - Kickstart ROM - Memoria RAM: 512 KBytes - 3 Chip custom per DMA, Video, Audio, I/O - 1 Floppy Disk Driver da 3 1/2", 880 KBytes - Porta seriale RS232C - Porta parallela Centronics

Videomaster 2995 - L. 1.200.000

Desk Top Video - Sistema per elaborazini video semiprofessionale composto da genlock, digitalizzatore e alloggiamento per 3 drive A2010 - Ingressi videocomposito (2), RGB - Uscite Videocomposito, RF, RGB + sync -

Floppy Disk Driver A 1010 - L. 335.000

Floppy Disk Driver - Drive esterno da 3 1/2" - Capacità 880 KBytes - Collegabile a tutti i modelli della linea Amiga, alla scheda A2088 e al PC1

Floppy Disk Drive A 2010 - L. 280.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" - Capacità 880 KBytes - Collegabile ad Amiga 2000

Hard Disk A 590 - L. 1.750.000

Hard Disk+Controller+RAM - Scheda Controller - Hard Disk da 3 1/2" 20 MBytes - 2 MBytes "fast" RAM - Collegabile all'Amiga 500

Scheda Janus A 2088 + A 2020 - L. 1.050.000

Scheda Janus XT+Floppy Disk Drive da 5 1/4", 360 KBytes - Scheda Bridgeboard per compatibilità MS-DOS (XT) in Amiga 2000 - Microprocessore Intel 8088 - Coprocessore matematico opzionale Intel 8087

A2286+A2020 - L. 1.985.000

Scheda Janus AT+Floppy Disk Drive da 5 1/4", 1.2 MBytes - Scheda Bridgeboard per compatibilità MS-DOS (AT) in Amiga 2000 - Microprocessore Intel 80287 - Clock 8 MHz - RAM: 1 MBytes on-board - Floppy Disk Controller on-board - Floppy Disk Driver disegnato per l'installazione all'interno dell'Amiga 2000 -

Scheda A2620 - L. 2.700.000

Scheda Processore Alternativo 32 bit - Scheda per 68020 e Unix - Microprocessore Motorola MC68020 - Coprocessore matematico Motorola MC68881 (opzionale MC68882)

Scheda A Unix - L. 3.250.000

Sistema Operativo AT&T Unix System V Release 3 - Per Amiga 2000 con scheda A2620 e Hard Disk 100 MBytes

Hard Disk A2092+PC5060 - L. 1.020.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+2092 - L. 1.240.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+A2094 - L. 1.900.000

Stesse caratteristiche del kit A2092 ma con disco da 40 MBytes

Espansione di memoria A2058 - L. 1.149.000

Espansione di memoria - Scheda di espansione per Amiga 2000 - Fornita con 2 MBytes "fast" RAM, espandibile a 4 o 8 MBytes

Scheda Video A2060 - L. 165.000

Modulatore video - Scheda modulatore video interna per Amiga 2000 - Uscite colore e monocromatica - Si inserisce nello slot video dell'Amiga 2000

Genlock Card A2301 - L. 420.000

Genlock - Scheda Genlock semiprofessionale per Amiga 2000 - Permette di miscelare immagini provenienti da una sorgente esterna con immagini provenienti dal computer

Professional Video Adapter Card A2351 - L. 1.500.000

Professional Video Adapter - Scheda Video Professionale per Amiga 2000 (B) - Genlock qualità Broadcast - Frame Grabber - Digitalizzatore - Include software di controllo per la gestione interattiva
(Disponibile da maggio '89)

A501 - L. 300.000

Espansione di memoria - Cartuccia di espansione di memoria da 512 KBytes per A500

A520 - L. 45.000

Modulatore RF - Modulatore esterno A500 - Permette di connettere qualsiasi televisore B/N o colori ad Amiga 500

A Scart - L. 27.000
Cavo di collegamento A500/A2000 con connettore per televisione SCART

Monitor a colori 1084 - L. 595.000
Monitor a colori ad alta risoluzione - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor a colori 2080 - L. 770.000
Monitor a colori ad alta risoluzione e lunga persistenza - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Frequenza di raster 50 Hz - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor Monocromatico A2024 - L. 1.235.000
Monitor monocromatico a fosfori "bianco-carta" - Turbo 14" antiriflesso - (Disponibile da marzo '89)

PC60/40 - L. 7.812.000
Microprocessore Intel 80386 - Coprocessore matematico opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 funzioni - Sistema Operativo MS-DOS 3 2.1 - Interprete GW-Basic

PC60/40C - L. 8.127.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 60/80 - L. 10.450.000
Microprocessore Intel 80386 - Coprocessore opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Memoria RAM: 2.5 MBytes - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Floppy Disk Drive opzionale da 3 1/2", 1.44 MBytes - 1 Hard Disk da 80 MBytes - 2 Porte parallele Centronics - Mouse video EGA (compatibile MDA - Hercules - CGA). Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Ambiente Operativo Microsoft Windows/386 - Interprete GW-Basic

PC60/80C - L. 10.700.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC40/20 - L. 4.100.000
Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/20C - L. 4.350.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 40/40 - L. 5.285.000
Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/40C - L. 5.535.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

1352 - L. 78.000
Mouse - Collegabile con Microsoft Bus Mouse - Collegabile direttamente a PC1, PC10/20 - III, PC40 - III

PC910 - L. 355.000
Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" per PC10/20-I-II-III - Capacità 360 o 720 KBytes selezionabile tramite "config. sys" - Corredo di telaio di supporto per l'installazione in un alloggiamento per un drive da 5 1/4" - Interfaccia identica ai modelli da 5 1/4"

PC1 - L. 995.000
Microprocessore Intel 8088 - 1 Floppy Disk Drive da 5 1/4" - Porta seriale RS232C - Porta parallela Centronics - - Monitor monocromatico 12" - Tastiera 84 tasti - Sistema Operativo MS-DOS 3.2 - Interprete GW-Basic

PCEXP1 - L. 640.000
PC Expansion Box - Box esterno di espansione per PC 1 - Alimentatore aggiuntivo incluso - Contiene 3 Slot di Espansione compatibili Ibm XT - Alloggiamento per Hard Disk da 5 1/4" - Si posiziona sotto il corpo del PC1 e viene collegato tramite degli appositi connettori

PC10-III - L. 1.360.000
Microprocessore Intel 8088 Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - Memoria RAM: 640 KBytes - 2 Floppy Disk Drive da 5 1/4", 360 KBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC10-IIIC - L. 1.675.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC20-III - L. 2.095.000
Microprocessore Intel 8088 - Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - 1/4", 360 KBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse) - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC20-IIIC - L. 2.410.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

Nuovo C64 - L. 325.000
Nuovo Personal Computer CPU 64 KBytes RAM - Vastissima biblioteca software disponibile - Porta seriale Commodore - Porta registratore per cassette - Porta parallela programmabile -

C128D - L. 895.000
Personal Computer CPU 128 KBytes RAM espandibile a 512 KBytes - ROM 48 KBytes - Basic 7.0 - Tastiera separata - Funzionante in modo 128,64 o CP/M 3.0 - Include floppy disk drive da 340 KBytes

Floppy Disk Drive 1541 II - L. 365.000
Floppy Disk Drive - Floppy Disk Drive da 5 1/4" singola faccia - Capacità 170 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

Floppy Disk Dirve 1581 - L. 420.000
Floppy Disk Drive da 3 1/2" doppia faccia - Capacità 800 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

1530 - L. 55.000
Registratore a cassette per C64, C128, C128D

Accessori per C64 - 128D
1700 - Espansione di memoria - Cartuccia di espansione di memoria a 128 KBytes per C128 - **L. 170.000**

1750 - Espansione di memoria - Cartuccia di espansione di memoria 512 KBytes per C128 - **L. 245.000**

1764 - Espansione di memoria - Cartuccia di espansione di memoria a 256 KBytes per C64 - Fornita di alimentatore surdimensionato - **L. 198.000**

16499 - Adattatore Telematico Omologato - Collegabile al C64 - Permette il collegamento a Videotel, P.G.E. e banche dati **L. 149.000**

1399 - Joystick - Joystick a microswitch con autofire - **L. 29.000**

1351 - Mouse - Mouse per C64, C128, C128D - **L. 72.000**

Monitor Monocromatico 1402 - L. 280.000
Monitor monocromatico a fosfori "bianco-carta" - Turbo 12" antiriflesso - Ingresso TTL - Compatibile con tutta la gamma PC

Monitor Monocromatico 1404 - L. 365.000
Monitor monocromatico a fosfori ambra - Turbo 14" antiriflesso a schermo piatto - Ingresso TTL - Compatibile con tutta la gamma PC - Base orientabile

Monitor Monocromatico 1450 - L. 470.000
Monitor monocromatico BI-SYNC a fosfori "bianco-carta" - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Monitor a colori 1802 - L. 445.000
Monitor a colori - Turbo 14" - Collegabile a C64, C128, C128D

Monitor monocromatico 1900 - L. 199.000
Monitor monocromatico a fosfori verdi - Turbo 12" antiriflesso - Ingresso videocomposito - Compatibile con tutta la gamma Commodore

Monitor a colori 1950 - L. 1.280.000
Monitor a colori BI-SYNC alta risoluzione - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Stampante MPS 1230 - L. 465.000
Stampante a matrice di punti - Testina a 9 aghi - 120 cps - Bidirezionale - 80 colonne - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

MPS 1230R - L. 19.000
Nastro per stampante

Stampante MPS 1500C - L. 495.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia parallela Centronics - Compatibile con la gamma Amiga e PC

MPS1500R - L. 37.000
Nastro a colori per stampante

MPS1500R - L.37.000
Nastro a colori per stampante

Stampante MPS 1550C - L. 575.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

I COMMODORE POINT

LOMBARDIA

Milano
• AL RISPARMIO - V.LE MONZA 204
• BCS - VIA MONTEGANI 11
• BRAHA A. - VIA PIER CAPPONI 5
• E.D.S. - C.SO PORTA TICINESE 4
• FAREF - VIA A. VOLTA 21
• FLOPPERIA - V.LE MONTENERO 31
• GBC - VIA CANTONI 7 - VIA PETRELLA 6
• GIGLIONI - V.LE LUIGI STURZO 45
• L'UFFICIO 2000 - VIA RIPAMONTI 213
• LOGITEK - VIA GOLGI 60
• LU - MEN - VIA SANTA MONICA 3
• MARCUCCI - VIA F.LLI BRONZETTI 37
• MELCHIONI - VIA P. COLLETTA 37
• MESSAGGERIE MUSICALI - GALLERIA DEL CORSO 2
• NEWEL - VIA MAC MAHON 75
• PANCOMMERZ ITALIA - VIA PADOVA 1
• SUPERGAMES - VIA VITRUVIO 38
• 68000 E DINTORNI - VIA WASHINGTON 91

Provincia di Milano
• GINO FERRARI CENTRO HI-FI - VIA MADRE CABRINI 44 - S. ANG. LODIGIANO
• F.LLI GALIMBERTI - VIA NAZIONALE DEI GIOVI 28/36 - BARLASSINA
• TECNOLUX - VIA PIETRO NENNI 5 - BERNATE TICINO
• OGGIONI & C. - VIA DANTE CESANA 27 - CARATE BRIANZA
• AL RISPARMIO - VIA U. GIORDANO 57 - CINISELLO BALSAMO
• GBC - V.LE MATTEOTTI 66 - CINISELLO BALSAMO
• CASA DELLA MUSICA - VIA INDIPENDENZA 21 - COLOGNO MONZESE
• PENATI - VIA VERDI 28/30 - CORBETTA
• EPM SYSTEM - V.LE ITALIA 12 - CORSICO
• P.G. OSTELLARI - VIA MILANO 300 - DESIO
• CENTRO COMPUTER PANDOLFI - VIA CORRIDONI 18 - LEGNANO
• COMPUTEAM - VIA VECELLIO 41 - LISSONE
• M.B.M. - C.SO ROMA 112 - LODI
• L'AMICO DEL COMPUTER - VIA CASTELLINI 27 - MELEGNANO
• BIT 84 - VIA ITALIA 4 - MONZA
• IL CURSORE - VIA CAMPO DEI FIORI 35 - NOVATE MIL.
• I.C.O. - VIA DEI TIGLI 14 - OPERA
• R & C ELGRA - VIA SAN MARTINO 13 - PALAZZOLO MIL.
• ESSEGIEMME SISTEMI SAS - VIA DE AMICIS 24 - RHO
• TECNO - CENTRO - VIA BARACCA 2 - SEREGNO
• NIWA HARD&SOFT - VIA B. BUOZZI 94 - SESTO SAN GIOV.
• COMPUTER SHOP - VIA CONFALONIERI 35 - VILLASANTA
• ACTE - VIA B. CREMIGNANI 13 - VIMERCATE
• IL COMPUTER SERVICE SHOP - VIA PADANA SUPERIORE 197 - VIMODRONE

Bergamo
•D.R.B. - VIA BORGO PALAZZO 65
• TINTORI ENRICO &C. - VIA BROSETA 1
• VIDEO IMMAGINE - VIA CARDUCCI c/o CITTA' DI MERCATO

Provincia di Bergamo
BERTULEZZI GIOVANNI - VIA FANTONI 48 - ALZANO LOMBARDO
• COMPUTER SHOP - VIA VITTORIO VENETO 9 - CAPRIATE SAN GERVASIO
• B M R - VIA BUTTARO 4/T - DALMINE
• MEGABYTE 2 - VIA ROMA 61/A - GRUMELLO
• OTTICO OPTOMETRISTA ROVETTA - P.ZZA GARIBALDI 6 - LOVERE
• COMPUTER POINT - VIA LANTIERI 52 - SARNICO
• A.B. INFORMATICA - STRADA STATALE CREMASCA 66 - URGNANO

Brescia
• MASTER INFORMATICA - VIA F.LLI UGONI 10/B

PROVINCIA DI BRESCIA
• MISTER BIT - VIA MAZZINI 70 - BRENO
• CAVALLI PIETRO - VIA 10 GIORNATE 14 BIS - CASTREZZATO
• VIETTI GIUSEPPE - VIA MILANO 1/B - CHIARI
• MEGABYTE - P.ZZA MALUEZZI 14 - DESENZANO DEL GARDA
• BARESI RINO &C. - VIA XX SETTEMBRE 7 - GHEDI
• INFO CAM - VIA PROVINCIALE 3 - GRATACASOLO
• "PAC-LAND" di GARDONI - CENTRO COM.le - LA CASA DI MARGHERITA D'ESTE - VIA GIORGIONI 21

Como
• IL COMPUTER - VIA INDIPENDENZA 90
• 2M ELETTRONICA - VIA SACCO 3

Provincia di Como
• ELTRON - VIA IV NOVEMBRE 1 - BARZANO'
• DATA FOUND - VIA A. VOLTA 4 -ERBA
• CIMA ELETTRONICA - VIA L. DA VINCI 7 - LECCO
• FUMAGALLI - VIA CAIROLI 48 -LECCO
• RIGHI ELETTRONICA - VIA G. LEOPARDI 26 - OLGIATE COMASCO

Cremona
• MONDO COMPUTER - VIA GIUSEPPINA 11/B
• PRISMA - VIA BUOSO DA DOVARA 8
• TELCO - P.ZZA MARCONI 2/A

Provincia di Cremona
• ELCOM - VIA IV NOVEMBRE 56/58 - CREMA
• EUROELETTRONICA - VIA XX SETTEMBRE 92/A - CREMA

Mantova
• COMPUTER CANOSSA - GAL. FERRI 7
• 32 BIT - VIA C. BATTISTI 14
• ELET. di BASSO - V.LE RISORGIMENTO 69

Provincia di Mantova
• CLICK - ON COMPUTER - S.S. GOITESE 168 - GOITO

Pavia
• POLIWARE - C.SO C. ALBERTO 76
• SENNA GIANFRANCO - VIA CALCHI 5

Provincia di Pavia
• A. FERRARI - C.SO CAVOUR 57 - MORTARA
• LOGICA MAINT - V.LE M.TE GRAPPA 32 - VIGEVANO
• M. VISENTIN - C.SO V. EMANUELE 76 - VIGEVANO

Sondrio
• CIPOLLA MAURO - VIA TREMOGGE 25

Provincia di Sondrio
• FOTONOVA - VIA VALERIANA 1 - S.PIETRO DI BERBENNO

Varese
• ELLE - EFFE - VIA GOLDONI 35
• IL C.TRO ELET. - VIA MORAZZONE 2
• SUPERGAMES - VIA CARROBBIO 13

Provincia di Varese
• BUSTO BIT - VIA GAVINANA 17-BUSTO A.
• MASTER PIX - VIA S.MICHELE 3 - BUSTO A.
• PUNTO UFFICIO - VIA R.SANZIO 8 - GALLARATE
• GRANDI MAGAZZINI BOSSI - VIA CLERICI 196 - GERENZANO
• J.A.C. - C.so MATTEOTTI 38 - SESTO C.

PIEMONTE

Alessandria
• BIT MICRO - VIA MAZZINI 102
• SERV. INFOR. - VIA ALESSANDRO III 47

Provincia di Alessandria
• SONY ITALIANA - VIA G. MANARA 7 - CASALE MONFERRATO
• SGE ELETTRONICA - VIA BANDELLO 19 - TORTONA
• COMPUTER TEMPLE - VIA F. CAVALLOTTI 13 - VALENZA

Asti
• ASTI GAMES - C.SO ALFIERI 26
• RECORD - C.SO ALFIERI 166/3 (Galleria Argenta)

Cuneo
• ROSSI COMPUTERS - C.SO NIZZA 42

Provincia di Cuneo
• PUNTO BIT - C.SO LANGHE 26/C - ALBA
• BOSETTI - VIA ROMA 149 - FOSSANO
• COMPUTERLAND - VIA MAZZINI 30/32 - SALUZZO

Novara
• PROGRAMMA 3 - V.LE BUONARROTI 8
• PUNTO VIDEO - C.so RISORGIMENTO 39/B

Provincia di Novara
• COMPUTER - VIA MONTE ZEDA 4 -ARONA
• ALL COMPUTER - C.SO GARIBALDI 106 - BORGOMANERO
• S.P.A. - C.SO DISSEGNA 21/BIS - DOMODOSSOLA
• ELLIOTT COMPUTER SHOP - VIA DON MINZONI 32 - INTRA
• TRISCONI VALERIA - VIA MAZZINI 90 - OMEGNA

Torino
• ABA ELETTRONICA - VIA C. FOSSATI 5/P
• ALEX COMPUTER E GIOCHI - C.SO FRANCIA 333/4
• COMPUTER HOME - VIA SAN DONATO 46/D
• COMPUTING NEW - VIA M. POLO 40/E
• C.D.M. ELETTR. - VIA MAROCHETTI 17
• DE BUG - C.SO V. EMANUELE II 22
• DESME UNIVERSAL - VIA S.SECONDO 95
• FDS ALTERIO - VIA BORGARO 86/D
• IL COMPUTER - VIA N. FABRIZI 126
• MICRONTEL - C.SO D. degli ABRUZZI 28
• PLAY GAMES SHOP - VIA C. ALBERTO 39/E
• RADIO TV MIRAFIORI - C.SO UNIONE SOVIETICA 381
• SMT ELETTRONICA - VIA BIBIANA 83/bis

Provincia di Torino
• PAUL E CHICO VIDEOSOUND - VIA V.EMANUELE 52 - CHIERI
• BIT INFORMATICA - VIA V. EMANUELE 154 - CIRIE'
• HI - FI CLUB - C.SO FRANCIA 92C - COLLEGNO
• MISTER PERSONAL - VIA CATTANEO 52 - FAVRIA
• I.C.S. - VIA TORINO 73 - IVREA
• DAG - VIA I MAGGIO 40 - LUSERNA S. GIOVANNI
• EUREX - C.SO INDIPENDENZA 5 - RIVAROLO CANAVESE
• DIAM INFORMATICA - C.SO FRANCIA 146 bis - RIVOLI
• FULLINFORMATICA - VIA V.VENETO 25 - RIVOLI
• GAMMA COMPUTER - VIA CAVOUR 3A-3B - SET.TORINESE

Vercelli
•ELETTROGAMMA - C.SO BORMIDA 27 ang. V.Montanara
• ELETTRONICA - STRADA TORINO 15

Provincia di Vercelli
• C.S.I. TEOREMA - VIA LOSANA 9 - BIELLA
• SIGEST - VIA BERTODANO 8 - BIELLA
• REMONDINO FRANCO - VIA ROMA 5 - BORGOSESIA
• FOTOSTUDIO TREVISAN - VIA XXV APRILE 24/B - COSSATO
• STUDIO FOTOGRAFICO IMARISIO - P.ZZA M. LIBERTA' 7 - TRINO

VENETO

Belluno
• UP TO DATE - VIA V. VENETO 43

Provincia di Belluno
• GUERRA COMPUTERS - V.LE MAZZINI 10/A - FELTRE

Padova
• BIT SHOP - VIA CAIROLI 11
• COMPUMANIA - VIA T. CAMPOSANPIERO 37
• D.P.R. DE PRATO R. - V.LO LOMBARDO 4
• G.F. MARCATO - VIA MADONNA DELLA SALUTE 51/53
• SARTO COMPUTER - VIA ARMISTIZIO 79

Provincia di Padova
• COMPUTER SERVICE - BORGO TREVISO 150 - CITTADELLA

Treviso
• BIT 2000 - VIA BRANDOLINI D'ADDA 14
• GUERRA EGIDIO & C. - V.LE CAIROLI 95

Provincia di Treviso
• DE MARIN COMPUTERS - VIA MATTEOTTI 142 - CONEGLIANO
• SIDESTREET - VIA SALVO D'ACQUISTO 8 - MONTEBELLUNA
• FALCON ELETTROAUDIOVIDEO - VIA TERRAGGIO 116 - PREGANZIOL

Venezia
• GUERRA EGIDIO & C. - VIA BISSUOLA 20/A - MESTRE
• TELERADIO FUGA - SAN MARCO 3457

Provincia di Venezia
• GUERRA EGIDIO & C. - VIA VIZZOTTO 29 - SAN DONA' DI PIAVE
• REBEL - VIA F. CRISPI 10 - SAN DONA' DI PIAVE

Verona
• CASA DELLA RADIO - VIA CAIROLI 10
• TELESAT - VIA VASCO DE GAMA 8

• Provincia di Verona
• UBER - CP 0363(RAG.SOC. DERTA) - VIA MASCAGNI 31 -CASTEL D'AZZANO
• FERRARIN - VIA DEI MASSARI 10 - LEGNAGO
• COMPUTERS CENTER - VIA CANTORE 26 - VILLAFRANCA

Vicenza
• ELET. BISELLO - V.LE TRIESTE 427/429
• SCALCHI MARKET - VIA CA' BALBI 139

Provincia di Vicenza
• SCHIAVOTTO - VIA ZANELLA 21 - CAVAZZALE
• GUERRA E. & C. - V.LE DELLE INDUSTRIE - MONTECCHIO MAGGIORE

FRIULI VENEZIA GIULIA

Gorizia
• E.CO. ELETTRONICA - VIA F.LLI COSSAR 23

Trieste
• AVANZO GIACOMO - P.ZZA CAVANA 7
• COMPUTER SHOP - VIA P. RETI 6
• COMPUTIGI - VIA XX SETTEMBRE 51
• CTI - VIA PASCOLI 4

Udine
• MOFERT 2 - VIA LEOPARDI 21
• R.T. SISTEM UDINE - VIA L. DA VINCI 99

Provincia di Udine
• IL PUNTO ELETTRONICO - VIA VENDRAMIN 184 - LATISANA
• IDRENO MATTIUSSI &C. - VIA LICINIANA 58 - MARTIGNACCO

TRENTINO ALTO ADIGE

Bolzano
• COMPUTER POINT - VIA ROMA 82/A
• MATTEUCCI PRESTIGE - VIA MUSEO 54

Provincia di Bolzano
• RADIO MAIR-ELECTRO - VIA CENTRALE 70 - BRUNICO
• ELECTRO RADIO HENDRICH - VIA DELLE CORSE 106 - MERANO
• ERICH KONTSCHIEDER - PORTICI 313 - MERANO
• ELECTRO TAPPEINER - P.ZZA PRINCIPALE 90 - SILANDRO

Trento
• CRONST - VIA G. GALILEI 25

Provincia di Trento

• AL RISPARMIO - C.SO VERONA 138 - ROVERETO

LIGURIA
Genova
• ABM COMPUTER - P.ZZA DE FERRARI 24 rosso
• CAPRIOTTI G. - IA MAMIANI 4r - SAMPIERDARENA
• C.tro ELET. - VIA CHIARAVAGNA 10 R - VIA SESTRI 69R
• COM.le SOTTORIPA - VIA SOTTORIPA 115/117
• FOTOMONDIAL - VIA DEL CAMPO 3-5-9-11-13 r
• LA NASCENTE - VIA SAN LUCA 4/1
• PLAY TIME - VIA GRAMSCI 3/5/7 rosso
• RAPPR-EL - VIA BORGORATTI 23 R
Imperia
• CASTELLINO - VIA BELGRANO 44
Provincia di Imperia
• CENTRO HI-FI VIDEO - VIA DELLA REPUBBLICA 38 -SANREMO
• CASTELLINO - VIA GENOVA 48 - VENTIMIGLIA
La Spezia
• I.L. ELETTRONICA - VIA V. VENETO 123
Provincia di La Spezia
• I.L. ELETTRONICA - VIA AURELIA 299 - FORNOLA DI VEZZANO
Savona
• CASTELLINO - C.SO TARDY E BENECH 101
Provincia di Savona
• CELESIA ENZA - VIA GARIBALDI 144 - LOANO

EMILIA
Bologna
• EUROELETTRICA - VIA RANZANI 13/2
• MINNELLA ALTA FEDELTA' - VIA MAZZINI 146/2
• MORINI & FEDERICI - VIA MARCONI 28/C
• STERLINO - VIA MURRI 73/75
Provincia di Bologna
• S.C. COMPUTERS - VIA E. FERMI 4 - CASTEL SAN PIETRO
• S.P.E. INFORMATICA - VIA DI MEZZO PONENTE 385 - CREVALCORE
• ARCHIMEDE SISTEMI - VIA EMILIA 124 - S. LAZZARO DI SAVENA
Modena
• CO - EL - VIA CESARI 7
• ORSA MAGGIORE - P.ZZA MATTEOTTI 20
• VIDEO VAL WILLY COMPUTERS - VIA CANALETTO 223
Provincia di Modena
• NEW MEDIA SYSTEM - VIA ROMA 281 - SOLIERA
Parma
• BABARELLI G. - VIA B. PARENTE 14/A/B
Provincia di Parma
• PONGOLINI - VIA CAVOUR 32 - FIDENZA
Piacenza
• COMPUTER LINE - VIA G. CARDUCCI 4
• DELTA COMPUTER - VIA M. DELLA RESISTENZA 15/G

TEGGIO EMILIA
• COMPUTERLINE - VIA SAN ROCCO 10/C
• POOL SHOP - VIA EMILIA S. STEFANO 9/C
Provincia di Reggio Emilia
• MACCHIONI - VIA STATALE 467 - CASALGRANDE
ROMAGNA
Ferrara
• BUSINESS POINT - VIA CARLO MAYER 85
Forlì
• COMPUTER VIDEO CENTER - VIA CAMPO DI MARTE 122
Provincia di Forlì
• TOP BIT - VIA VENETO 12 - FORLIMPOPOLI
• COMPUTER HOUSE - V.LE TRIPOLI 193/D - RIMINI
• EASY COMPUTER - VIA LAGOMAGGIO 50 - RIMINI

REPUBBLICA S. MARINO
Ravenna
• COMPUTER HOUSE - VIA TRIESTE 134
Provincia di Ravenna
• ARGNANI - P.ZZA DELLA LIBERTA' 5/A - FAENZA
• ELECTRON INFORMATICA - VIA F.LLI CORTESI 17 - LUGO
• P.L.Z. INFORMATICA - P.ZZA SERCOGNANI 6 - FAENZA

TOSCANA
Arezzo
• DELTA SYSTEM - VIA PIAVE 13
Firenze
• ATEMA - VIA BENEDETTO MARCELLO 1a-1b
• ELETTRONICA CENTOSTELLE - VIA CENTO STELLE 5/a-b
• HELP COMPUTER - VIA DEGLI ARTISTI 15-A
• TELEINFORMATICA TOSCANA -VIA BRONZINO 36
Provincia di Firenze
• WAR GAMES - VIA R. SANZIO 126/A - EMPOLI
• NEW EVM COMPUTER - VIA DEGLI INNOCENTI 2 - FIGLINE VALDARNO
• C.tro INFOR. - VIA ZNOJMO 41 - PONTASSIEVE
• COSCI F.LLI - VIA ROMA 26 - PRATO
• BARBAGLI C. ELET. - VIA F. BONI 80 - PRATO
Grosseto
• COMPUTER SERVICE - VIA DELL'UNIONE 7
Livorno
• ETA BETA - VIA SAN FRANCESCO 30
• FUTURA 2 - VIA CAMBINI 19
Provincia di Livorno
• PUNTO ROSSO - VIA BARONTINI 28 - PIOMBINO
Provincia di Lucca
• IL COMPUTER - V.LE COLOMBO 216 - LIDO DI CAMAIORE
• SANTI VITTORIO - VIA ROMA 23 - S. ROMANO GARFAGNANA
• TOP GAMES - VIA S. ANDREA 122 - VIAREGGIO
Massa
• EURO COMPUTER - P.ZZA G. BERTAGNINI 4
Carrara
• RADIO LUCONI - VIA ROMA 24/B
Pisa
• ELECTRONIC SERVICE - VIA DELLA VECCHIA TRANVIA 10
• PUCCINI S.- CP 1199 (RAG.SOC. MAREX) - VIA C.CAMMEO 64
• TONY HI-FI - VIA CARDUCCI
Provincia di Pisa
• M.C. INFORMATICA - VIA DEL CHIESINO 4 - PONTEDERA (PI)
Pistoia
• ELECTRONIC SHOP - VIA DEGLI SCALZI 3
Provincia di Pistoia
• ZANNI &C. - C.SO ROMA 45 - MONTECATINI T.

Siena
• R. BROGI - P.ZZA GRAMSCI 28
• VIDEO MOVIE - VIA GARIBALDI 17
Provincia di Siena
• ELETTRONICA di BIFOLCHI - VIA DI GRACCIANO NEL CORSO 111 - MONTEPULCIANO

LAZIO
• CENTRO INF. - D.R.R. srl - TEL. 06-5565672

UMBRIA
Perugia
• MIGLIORATI - VIA S. ERCOLANO 3-10
Provincia di Perugia
• COMPUTER STUDIO'S - VIA IV NOVEMBRE 18/A - BASTIA UMBRA
• WARE - VIA DEI CASCERI 31 - CITTA'DI CASTELLO
Terni
• CGS SOFTWARE HOUSE - VIA DONIZETTI 71/A

BASILICATA
Matera
• G. GAUDIANO ELECTRONICS - VIA ROMA ang. XX SETTEMBRE 1

PUGLIA
Bari
• ARTEL - VIA GUIDO D'ORSO 9
• COMPUTER'S ARTS - V.LE MEUCCI 12/B
• PAULICELLI S. & F. - VIA FANELLI 231/C
Provincia di Bari
• F. FAGGELLA - C.SO GARIBALDI 15 - BARLETTA
• G.FAGGELLA - P.ZZA D'ARAGONA 62A - BARLETTA
• LONUZZO G. - VIA NIZZA 21 - CASTELLANA
• TECNOUFF. - VIA RICASOLI 54 - MONOPOLI
• TANGORRA N. - C.SO V.EMANUELE 130/B - TRIGGIANO
Brindisi
• MARANGI E NICCOLI - VIA PROV. SAN VITO 165
Provincia di Brindisi
• MILONE G. - VIA S.F. D'ASSISI 219 - FRANCAVILLA FONTANA
Foggia
• BOTTICELLI G. - VIA SAV POLLICE 2
• E.C.I. COMPUTER - VIA ISONZO 28
• LA TORRE - V.LE MICHELANGELO 185
Provincia di Foggia
• IL DISCOBOLO - VIA T. SOLIS 15 - SAN SEVERO
Lecce
• BIT - VIA 95 REGG.NTO FANTERIA 87/89
Provincia di Lecce
• TECNO UFFICIO - P.ZZA GIOVANNI XXIII 10 - GALLIPOLI
• CEDOK INFORMATICA - VIA UMBERTO I 116 - TRICASE
Taranto
• ELETTROJOLLY C.tro - VIA DE CESARE 13
• TEA - TEC. ELET. AV. - VIA R. ELENA 101

CAMPANIA
Provincia di Avellino
• FLIP FLOP - VIA APPIA 68 - ATRIPALDA
Benevento
• E.CO. INF. - VIA PEPICELLI 21/25
Caserta
• ENTRY POINT - VIA COLOMBO 31
• O.P.C. - VIA G. M. BOSCO 24
Provincia di Caserta
• M.P. COMPUTER - VIA NAPOLI 30 - MADDALONI
• DAMIANO - C.SO V. EMANUELE 23 -ORTA DI ATELLA
• FUSCO B. - VIA NAPOLI 24 - VAIRANO PATERNORA (FRAZ. VAIRANO SCALO)
• LINEA CONTABILE - VIA OSPEDALE 72/76 - SESSA A. (CE)
Napoli
• BABY TOYS - VIA CISTERNA DELL'OLIO 5/BIS
• CASA MUSICALE RUGGIERO - P.ZZA GARIBALDI 74 (INT. STAZ. F.F. S.S.)
• C.tro ELET. CAMPANO - VIA EPOMEO 121
• CI.AN - GALLERIA VANVITELLI 32
• CINE NAPOLI - VIA S. LUCIA 93/95
• DARVIN - CALATA SAN MARCO 26
• GIANCAR 2 - P.ZZA GARIBALDI 37
• ODORINO - L.GO LALA 22 A-B
• R 2 - VIA F. CILEA 285
• SAGMAR - VIA S. LUCIA 140
• TOP VIDEO - TOP COMPUTER - VIA S. ANNA DEI LOMBARDI 12
• VIDEOFOTOMARKET - VIA S. BRIGIDA 19
Provincia di Napoli
• ELECTRONIC DAY - VIA DELLE PUGLIE 17 - CASORIA
• TUFANO - S.S. SANNITICA 87 KM 7 - CASORIA
• SOF SUD - V.LE EUROPA 59 - CASTEL/MARE DI STABIA
• ELETTRONICA 2000 - C.SO DURANTE 40 - FRATTAMAGGIORE
• SPADARO - VIA ROMANI 93 - MADONNA DELL'ARCO
• GATEWAY - VIA NAPOLI 68 - MUGNANO
• VISPINI & DI VUOLO - VIA A.ROSSI 4 - POMPEI
• SPY CASH & CARRY - P.ZZA ARENELLA 6/A - NAPOLI
• NUOVA INFORMATICA SHOP - VIA LIBERTA' 185/191 - PORTICI
• BASIC COMPUTER - C.SO GARIBALDI 34 - POZZUOLI
• V.C. - C.SO SECONDIGLIANO 562/B - SECONDIGLIANO
• F. ELETTRONICA - VIA SARNO 102 - STRIANO
• TECNO - VIA V. VENETO 48 - TORRE DEL GRECO
Salerno
• COMPUMARKET - VIA BELVEDERE 35
• COMPUTER MARKET - C.SO VITTORIO EMANUELE 23
Provincia di Salerno
• KING COMPUTER - VIA OLEVANO 56 - BATTIPAGLIA
• DIMER POINT - V.LE AMENDOLA 36 - EBOLI
• IACUZIO F. - VIA MUNICIPIO 14 - MERCATO SAN SEVERINO
• COMPUTER SERVICE - VIA L.DA VINCI 81 - SCAFATI

CALABRIA
Catanzaro
• C. & G. COMPUTER - VIA F. ACRI 28
• PAONE S. & F. - VIA F. ACRI 93/99
Provincia di Catanzaro
• COMPUTER HOUSE - VIA BOLOGNA (L.GO OSPEDALE) - CROTONE
• RIOLO F.LLI - VIA VENEZIA 1/7 - CROTONE
• ING. FUSTO S. - C.SO NICOTERA 99 - LAMEZIA TERME
Cosenza
• MAISON DE L'INFORMATIQUE - VIA PASQUALE ROSSI 34/C
• SIRANGELO COMP. - VIA N. PARISIO 25
Provincia di Cosenza
• HI-FI ALFANO G. - VIA BALDACCHINI 109 - AMANTIA
• ELIGIO ANNICCHIARICO &C. - VIA ROMA 21 - CASTROVILLARI
• ALFA COMPUTER - VIA NAZIONALE 341/A - CORIGLIANO SCALO
REGGIO CALABRIA
• CONTROL SYSTEM - VIA S.F DA PAOLA 49 D
• SYSTEM HOU. - VIA FIUME ang. PALESTINO 1
Provincia di Reggio Calabria
• COMPUTER SHOP - V.LE MATTEOTTI 36/38 - LOCRI
• PICIEFFE - C.SO F. S. ALESSIO 19 - TAURIANOVA
SICILIA
• CENTRO INF. - ITALSOFT SRL - TEL. 0935-696090

A MAGGIO
DECOLLA CON
JONATHAN